# Руководство по BDMS/4

ЛСФ ОАФ ИФВЭ

Протвино, 1997

# Введение

Система Управления Базами Данных (СУБД) BDMS (Berkeley Data Management System) задумана и создана в LBL (Lawrence Berkeley Laboratory) в 1972-1980 гг. как основа развивающейся информационной системы физики частиц PPDS (Particle Physics Data System). С помощью этой СУБД и ее последующих модификаций в нескольких лабораториях (LBL, RAL, ИФВЭ, CERN, ИТЭФ) создавались и сопровождаются фактографические массивы данных на разных ЭВМ.

Сложилось и длительное время действует международное информационное сотрудничество в области физики высоких энергий. Возникнув во времена недоразвитых СУБД содружество произвело для собственных нужд программное обеспечение решая задачи накопления данных, производство на основе отобранных данных печатной справочной продукции и, в меньшей степени, задачу непосредственного обслуживания пользователей. BDMS/2.2, сам явившийся репликой с еще более ранней системы SPIRES, во всех отношениях удовлетворяет потребности квалифицированного пользователя, администратора BDMS базы данных, однако при встрече со случайным пользователем демонстрирует несовременную скромность в общении. BDMS/4 — одна из попыток развить диалоговые возможности системы, сделать ее доступной, более удобной и сравнимой по возможностям с наиболее распространенными в настоящее время СУБД.

Данное руководство посвящено версии BDMS/4.0, Версия BDMS/4 машинно-зависима, ориентирована на использование под операционной системой VMS. Выбор такой ориентации объясняется широким распространением ЭВМ VAX под VMS в физических лабораториях.

Развитие BDMS/2.2 → BDMS/4.0 проведено без перестройки концептуальной схемы системы. В набор инструкций BDMS/4.0 включены все инструкции предыдущих версий. BDMS/4.0 поддерживает фиксированные в версии 2.2 схемы и форматы обмена данными между разными "экземплярами" PPDS.

Из других действующих "экземпляров" системы отметим версию BDMS развиваемую группой PDG(UK) (Particle Data Group) для ЭВМ IBM. Исходный BDMS/2 установлен на различных ЭВМ в лабораториях СССР, что позволило расширить советскую часть проекта PPDS, увеличить число его участников. PDG(US) используют машинно-зависимую систему SPIRES, структура данных в которой аналогична структуре данных BDMS.

В силу указанных причин PPDS в обозримое время будет продолжать опираться на BDMS как базовую СУБД, и этот факт стал основной причиной написания данного руководства для ее потенциальных пользователей.

BDMS — программа на языке FORTRAN (BDMS/2 — FORTRAN 4, BDMS/4 —

FORTRAN 77), которую можно разделить на две части:

1. **ЯДРО** системы: набор процедур оформления и реализации поиска в базе данных, обмена записями с памятью на диске, преобразования записи во внутреннее и внешнее представление, создания и поддержания инвертированных списков;

2. прикладные **ПРОЦЕССОРЫ** контроля ввода/вывода данных, их преобразования.

Процессоры пишутся на Фортране, и совместно с ядром системы образуют ЗАГРУЗОЧНЫЙ МОДУЛЬ (файл типа EXE). В PPDS столько различных загрузочных модулей сколько различных прикладных процессоров. Загрузочному модулю присваивается имя прикладного процессора включенного в него. Например:

| | |
|---|---|
| **DOC.EXE** | загрузочный модуль сопровождения документальных баз данных; |
| **RD.EXE** | загрузочный модуль, включающий прикладные процессоры контроля описания реакций в ФВЭ; |
| **BDMS.EXE** | загрузочный модуль, не имеющий прикладных процессоров контроля. |

В данном описании перенесение имени процессора контроля на загрузочный модуль отражено также в том, что и сам загрузочный модуль в дальнейшем называется ПРОЦЕССОРОМ. В тех случаях, когда может возникать неоднозначность дается необходимое уточнение.

Конструирование новой базы данных, как правило, включает в себя написание для нее прикладных процессоров, и создание специализированного загрузочного модуля. В рамках PPDS системы такой деятельностью занимается АДМИНИСТРАТОР базы данных, в функции которого входит также контроль за процессом сопровождения базы данных.

**Кому адресуется данное описание? Пользователи BDMS/4:**

1. физик-теоретик или экспериментатор, обращающийся к информационному массиву PPDS как к справочнику по физике высоких энергий. На этом уровне вполне достаточным для работы является описание КОМПАС-Букварь /1/ и описания структур записей основных PPDS баз данных.

2. администратор личной базы данных под управлением BDMS, формируемой из отобранных в стандартных базах данных PPDS записей с помощью стандартных для BDMS процедур. Кроме описания /1/ в этом случае потребуется освоить и данное описание[1].

3. администратор базы данных, использующий собственные процессоры контроля. Дополнительные источники информации: BDMS PROGRAMMER'S MANUAL, для версий BDMS/2, BDMS/4. На этом уровне от пользователя требуется знание FORTRAN-77.

[1] В системе PPDS это не теоретически возможный пользователь. Все ИНДЕКСАТОРЫ, т.е. персоны, пополняющие базы данных PPDS, являются как минимум пользователями такого уровня. Документ, прежде чем попадет в PPDS базу данных общего пользования проходит систему многократного контроля в микробазах индексаторов.

# Глава 1

# Структура данных

BDMS — система управления **базами** данных (БД), состоящими из **записей**, имеющих иерархическую структуру **элементов данных** (ЭД), по инвертированному списку которых осуществляется поиск.

База данных — файлы прямого доступа, размещенные на дисковом носителе. Вместе с файлами загрузочных модулей системы, вспомогательными файлами управления и контроля, файлами описаний и самоописаний файлы баз данных образуют **систему файлов** системы BDMS/4 отображенную в **дистрибутиве** PPDS.

Данная глава посвящена описанию перечисленных элементов структуры данных в BDMS/4.

## 1.1 База данных

Наличие различных загрузочных модулей-процессоров для различных типов информационных массивов явилось одной из причин дробления информационного банка PPDS на разные файлы: БАЗЫ ДАННЫХ. База данных BDMS — файлы с записями, содержащими информацию, отображающую какую-либо существенную часть предметной области системы PPDS.

Информация из публикации преобразуется индексатором во входной ДОКУМЕНТ системы, который вводится (ЗАГРУЖАЕТСЯ) в базу данных. Система обеспечивает хранение введенной в базу данных информации, пополнение ее, редактирование непосредственно в базе данных, удаление из базы данных, поиск в соответствии с запросом пользователя.

Для BDMS/4 проведено разделение баз данных на ОСНОВНЫЕ, открытые для доступа всем пользователям PPDS, и ПОЛЬЗОВАТЕЛЬСКИЕ, доступные только их хозяевам, размещенные в пользовательских директориях. Основные базы данных размещаются в специальной системной директории BDMS$BASE (что и определяет их как основные).

При вызове системы тип базы данных указывается в параметрах инструкции вызова. Параметр MAIN открывает доступ на запись к основной базе данных, USER — к пользовательской.

В BDMS/4 отсутствует надежная система защиты информации в базе данных. Основной источник возможной порчи базы данных: сбои ЭВМ во время ввода или редактирования данных администратором базы.

Администратор основной базы данных в качестве защитных мер может использовать:

1. введение запрета на несанкционированный ввод данных в базу данных, указанием системе директории, из которой ввод данных разрешен. По умолчанию ввод данных в основную базу данных разрешен отовсюду. Если запрет на несанкционированный ввод данных установлен, то при попытке пользователя войти в базу данных в моде доступа MAIN система печатает сообщение:

   ```
   Sorry. You have no privilege to use MAIN access to this base
   ```

2. использование параметра вызова системы запрещающего запись в базу данных. Такие параметры введены для обоих типов баз данных. Это READ для основной, и USRD для пользовательской. При таком уровне доступа система будет имитировать ввод и редактирование данных в основной базе данных, если эти процедуры понадобятся пользователю, но реальных записей не произведет;

3. система фиксирует начало и завершение реального переписывания в основной базе данных. База данных во время ее редактирования администратором закрыта для пользователя. Но система не позволит открыть на редактирование базу данных если ею кто-либо пользуется. При сбое доступ к базе данных автоматически прекращается;

4. в случае порчи данных администратором производится откат и ПЕРЕЗАГРУЗКА базы данных.

Инициирование всех СИМВОЛОВ и ЛОГИЧЕСКИХ ИМЕН необходимых для использования BDMS/4 будет произведено при первом использовании в текущем задании инструкции

**PPDS**

Рекомендуемое начало общения с системой — инструкция:

**PPDS ?**

По этой инструкции будет приведен список раскрытых для использования баз данных, и далее, по меню структурированное описание правил пользования BDMS, машинная версия препринта КОМПАС-Букварь /1/, или обзор системы файлов PPDS. Отметим, что символ "?" в синтаксисе PPDS-BDMS/4 означает вызов описания, соответствующего контексту, в котором он используется.

Инструкция:

**PPDS** {имя базы} (1)

открывает пользователю доступ к указанной базе данных. С этого момента пользователь использует командный язык системы BDMS до тех пор, пока не скомандует:

**STOP**

инструкция выхода из BDMS, или

**HOST**, "VMS инструкция"

для временного перехода из BDMS в VMS с последующим возвращением в BDMS по исполнении VMS инструкции.

Инструкция (1) в версии BDMS/4 действует "штатно" если:

а) набрана без ошибок, например, в имени базы данных;

б) администратор не закрыл доступ к базе данных (в этом случае печатается сообщение); в) если база данных не оказалась испорченной. В первом случае пользователю будет выведена на экран подсказка: какие базы данных доступны и как к ним обратиться. Подсказка выполнена в виде меню: список доступных баз данных. Выбор нужной осуществляется набором ее имени или части имени достаточной для того, чтобы избежать неоднозначности.

Инструкция:

**PPDS** {имя базы} ?

вызов краткого описания указанной базы данных.

В интерактивном режиме BDMS приглашает пользователя к общению титулом:

| {имя базы}:READ:EXEC> _ |
|---|

Знак _ в тексте данного описания отмечает положение на экране курсора при обращении системы к пользователю.

"EXEC" — имя основной подсистемы BDMS с которой начинается работа пользователя с базой данных.

"READ" — маркер уровня доступа к базе данных. При использовании макрокоманды PPDS возможен только такой уровень.

При первой встрече с титулом BDMS рекомендуется воспользоваться инструкцией:

??

по которой система выводит на экран список доступных команд с их кратким описанием. (ЗАМЕЧАНИЕ: часть команд, помеченных в описании "NOT YET IMPLEMENTED" принадлежат будущим версиям системы и не действуют в настоящей. Использование инструкции "??" предоставляет возможность быть в курсе новаций системы).

Для уточнения можно запросить у системы описание конкретной команды:

?{имя команды}

С соображениями администратора о том, зачем данная база создана, как пользоваться ее информацией можно познакомиться используя структурированное описание базы данных, вызываемое инструкцией:

**HELPBASE**

Если администратор не позаботился о создании такого описания, то пользователь будет об этом предупрежден при старте системы:

| NOT FOUND:<br>"{Имя файла, содержащего описание базы данных}" |
|---|

В BDMS/4 на тексты описания баз данных введен и действует стандарт. В соответствии с этим стандартом в тексте должны быть представлены:

1. желательно обе версии: английская и национальная;

2. описание предметной области, отраженной в базе данных;

3. структура данных, принятая в данной базе данных;

4. описания всех элементов данных;

5. ПРИМЕР(Ы) использования базы данных;

6. список ответственных за ее содержимое и их адреса.

В ПРИЛОЖЕНИИ HELP приводятся правила создания файла описаний базы данных.

BDMS/4 снабжен встроенной почтой, используемой для общения администраторов и пользователей BDMS. При наличии послания, BDMS при старте предупреждает об этом сообщением:

| |
|---|
| Mail box: {Имя файла с сообщением:}<br>Would you want to read mail for you? > _ |

которое не должно пугать. Ответ системе: YES означает, что пользователь желает ознакомиться с посланием, NO, или просто возврат каретки — просит оставить его в покое.

Все инструкции пользователя и вводимые им данные дублируются системой в выходном контрольном файле: {имя базы}.AUD (аудит-файл). От дублирования можно отказаться инструкцией:

**NOAUDIT**

и снова включить его инструкцией:

**AUDIT.**

При неинтерактивном режиме работы инструкции и данные дублируются также в другой контрольный файл: LOG. Туда же передается и информация о формальных ошибках в инструкциях или структуре вводимых данных, выявленных ядром системы или ее процессорами. От накопления LOG файла можно отказаться инструкцией:

**@/NOLOG.**

## 1.2 Запись в базе данных

База данных BDMS состоит из ЗАПИСЕЙ — единиц информации которыми производится обмен между системой и дисковым накопителем. Запись, как правило, логически обособляемая из всего информационного массива часть. Например, для документальных (библиографических) баз данных запись является библиографическим описанием оригинала: препринта, статьи, сообщения...

Терминология: Условимся называть записью базы данных (записью) информацию, размещенную в базе данных, ДОКУМЕНТОМ базы данных (документом) ту же информацию, выведенную из базы данных (на экран дисплея или в текстовый файл), или вводимую в базу данных. Т.е., запись — внутреннее представление документа, документ — внешнее представление записи. Выражения "запись документа" - обобщающая ссылка на процедуру ввода информации в базу данных, "документ записи" — вывода. Источник информации (статью, препринт, сообщение — определим как ОРИГИНАЛ).

Записи в базе данных размещаются последовательно, каждой соответствует УНИКАЛЬНЫЙ НОМЕР, (порядковый номер в базе). Номер этот не является логической принадлежностью записи. Возможна ситуация, при которой какому-либо номеру не будет соответствовать никакая запись (например, если ранее создававшаяся запись удалена из базы). В общем случае при перезагрузке базы данных номера записей меняются.

Номер записи в BDMS имеет имя: REC-ID. Выбрать запись из базы можно инструкциями:

**GET**,{n} (1)

где n — номер записи, или:

**FIND** REC-ID = {n}; ** (2)

В обоих примерах система выберет из базы данных n-ую запись и создаст ВЫБОРКУ — этим термином в BDMS обозначается список записей отобранный системой по инструкции пользователя из базы данных и идентифицируемый номером. При интерактивной работе система на приведенные выше инструкции ответит автоматически печатая ТАБЛИЦУ ВЫБОРОК:

```
    SET N.REC  COMMAND:
     1     1  :GET,1
 *   2     1  :FIND REC-ID=2; **
```

В этой таблице приведены: номер выборки (колонка SET), число записей в ней (колонка N.REC), инструкция, приведшая к созданию выборки (колонка COMMAND:). В безымянной первой колонке символом "*" отмечается ТЕКУЩАЯ ВЫБОРКА. Инструкции документирования записей, их удаления из базы данных, редактирования адресуются к записям текущей выборки. Обращение к конкретной записи в выборке осуществляется по ее порядковому номеру в выборке. Инструкция:

**SET**,{s}

делает текущей s-ую выборку в таблице. Если s оказывается больше чем номер последней выборки, например s=3 для приведенной в тексте таблицы, система предупреждает:

```
Warning. Invalid numerical value: 3 for NUMBER OF SETS
Used for control: MIN=1, MAX=2, Setted as default: 2
```

что означает, что система, опознав ошибку, все же ПРОИЗВЕЛА замену текущей выборки на последнюю в таблице.

Удалить ПОСЛЕДНЮЮ выборку из таблицы можно инструкцией:

**PURGE**,{s}

при этом s-ая становится текущей. Эта же инструкцмя, но без параметров используется для удаления ВСЕХ выборок:

**PURGE**

Количество записей в выборке "NREC" может быть от 0 (ПУСТАЯ ВЫБОРКА), до полного числа записей в базе. Пустая выборка равноправна с остальными для процедур манипулирующих выборками. При попытке обратиться к записям пустой выборки система выдаст предупреждение, зависимое от контекста.

Узнать объем базы данных в записях можно с помощью инструкции:

**FIND** REC-ID;**

Инструкция (2) в отличии от (1) — пример использования основной инструкции выбора записей из базы данных. Ее общий формат:

**FIND**_условие выбора **

(здесь и далее символ "_" означает обязательный пробел).

УСЛОВИЕ ВЫБОРА или ЗАПРОС в формате инструкции (3) формируется по правилам рассмотренным подробно в следующей главе. (см. 2.5).

## 1.3 Элемент данных в базе данных

Наименьшей логической единицей базы данных BDMS является ЭЛЕМЕНТ ДАННЫХ, (ЭД). Ядром системы BDMS/4 структура самого ЭД не рассматривается. При необходимости такую задачу решают прикладные процессоры.

Основные функции ЭД в базе данных:

1. разделение данных в записи на минимально значимые элементы, классификация данных;
2. объединение данных из различных записей в общей для них базе данных по общему признаку;
3. введение иерархического описания данных в документе (записи), структурирование данных;

Схематически соотношение между базой данных, записями в ней и элементами данных можно изобразить следующим образом:

Конструируя базу данных (все поле очерченное рамкой на рис.1.1.) администратор:

- определяет понятие "запись" в данной базе данных (на рисунке запись изображена горизонтальным рядом);
- вводит описание записи как упорядоченного набора логически выделяемых единиц информации - элементов данных (на рисунке — квадратики);
- фиксирует структуру записи введя значения атрибутов всех ее элементов данных.

ЭД из разных записей на рис.1.1 размещенные в одном столбце (например ЭД) для системы — одинаковые объекты, свойства которых задаются значениями атрибутов ЭД. Если среди них есть указывающий на КЛЮЧЕВУЮ роль данного ЭД в базе данных (такие на рисунке связаны вертикальной линией), то система физически объединит

Рис. 1.1:

такие ЭД, переписав из всех записей в общий инвертированный список — индекс ключевого ЭД.

Один и тот же ЭД может быть неоднократно представлен своими ГЕНЕРАЦИЯМИ в данной записи. Например, если содержимым ЭД является имя одного из авторов публикации, а запись — библиографическое описание публикации, то ЭД "АВТОР" в записи будет представлен столько раз, сколько соавторов содержит публикация. На рисунке подобная ситуация с несколькими ГЕНЕРАЦИЯМИ одного и того же ЭД обозначена связанной группой квадратиков.

Число генераций данного ЭД в записи может меняться от записи к записи от нуля (например ЭД$_1$ в записи 2) до любого количества, пока не сработает общее ограничение на размер записи.

### 1.3.1 Атрибуты элемента данных

*Наименование атрибута выделено жирным шрифтом. Рядом в скобках приводится понятное системе описание атрибута, в котором перед знаком "=" имя атрибута,* *между "=" и ";" — значение атрибута.*

**ИМЯ** (DE = {text};)

{text} — цепочка символов не включающая пробел, знак равенства, точку, точку с запятой. Имя элемента данных уникально, т.е. другие ЭД этой же базы данных не могут иметь такого имени. Имя ЭД используется при вводе документа в базу данных, при формулировании запроса. Длина имени ЭД в разумных пределах не ограничивается.

**СИНОНИМ** (SYN = {text};)

{text} — цепочка символов, формируемая по тем же правилам, что и имя ЭД. Имя и синоним к нему равноправны во внешнем представлении. То есть, в документе, формулах запроса, в инструкциях системе можно использовать как имя, так и синоним к нему. Во внутреннем представлении синоним зависим от имени ЭД. Это выражается в том, что при документировании записи системой будет использовано имя ЭД. Системой не ограничивается количество синонимов к имени ЭД.

**РОДИТЕЛЬ** (PAR = {text};)

Этот атрибут имени ЭД специфицирует в иерархической структуре записи старший элемент данных. Если родитель для ЭД не задан, то по умолчанию, родителем такого элемента данных является запись, такой ЭД называется элементом данных УРОВНЯ ЗАПИСИ. Элементы данных подчиненные ЭД уровня записи называются ЭД второго уровня, и так далее... Количество уровней не ограничено. Ссылка на родительский ЭД возможна только если сам родительский ЭД уже определен. Этим ограничением устраняется, в частности, возможность закольцевать связи между ЭД.

Выше нами уже рассматривался пример элемента данных с именем REC-ID, значением которого является номер записи в базе данных. В структуре данных BDMS этому ЭД принадлежит роль связующего элемента между записью, как целое, и составляющими ее элементами данных. Являясь обязательной принадлежностью записи этот ЭД не представлен в документе. ЭД обладающие таким свойством называются ВИРТУАЛЬНЫМИ. REC-ID — единственный в системе СИСТЕМНЫЙ ЭЛЕМЕНТ ДАННЫХ. Все остальные задаются при конструировании базы данных.

**ЗНАЧЕНИЕ** ( = {текст} ;)

Этот атрибут ЭД не имеет имени. Его свойства определяются значениями других атрибутов элемента данных: TY, IPROC, KEY, и др. (см. ниже).

Элемент данных представлен в записи своим значением. В документе имя и значение ЭД объединяются выражением, общая форма которого:

{имя}={значение};

Значение элемента данных в документе — это цепочка символов всегда завершаемая символом ";". При необходимости вставить символ ";" в текст значения его нужно использовать дважды. Пробелы в начале цепочки символов значения и в конце, перед ограничивающим символом ";" во внутреннем представлении игнорируются. Если требуется передать начальные или конечные пробелы во внутреннее представление, то используется символ ПРИНУДИТЕЛЬНОГО ПРОБЕЛА "_".

**ТИП значения элемента данных** (TY = {text};)

Этот атрибут описывает системе правило перевода значения ЭД из внешнего текстового представления во внутреннее представление в записи. Во внутреннем представлении значение ЭД может быть:

| | | |
|---|---|---|
| а) | текстовой строкой; | TY = CHAR; |
| б) | целочисленным вектором; | TY = INTE; |
| в) | вектором действительных чисел; | TY = REAL; |
| г) | вектором чисел двойной точности; | TY = DOUB; |
| д) | чисто узловые, не имеющие значения; | TY = NODE; |

ЭД любого типа можно присвоить нулевое, пустое значение:

{имя} = ;

Значения целочисленных ЭД во внешнем представлении должны быть серией цифр при необходимости предваряемых знаками "+" или "-". Оно может быть выражено в двоичном, восьмеричном, десятичном или шестнадцатеричном представлениях. База представления целого числа специфицируется следующей за числом буквой:

| | |
|---|---|
| B | двоичное, |
| O или Q | восьмеричное, |
| D или ничего | десятичное, |
| H или Z | шестнадцатеричное. |

В зависимости от используемой базы представления числа цифры изображаются символами от 0 до 9 и от A до F.

Значения действительных чисел или чисел двойной точности должны быть серией десятичных цифр предваряемых знаком "+" или "-". Они могут содержать десятичную точку и/или показатель десятичной степени выражаемый буквой "E" при целочисленной экспоненте со знаком.

Значения числовых типов (INTE, REAL, DOUB) в записи представлены соответствующими числами, а это значит, что их величина не должна превосходить максимально допустимой на ЭВМ.

Для 32-х разрядной машины это:

2147483647 для INTE

0.999999E38 REAL

0.999999999999E38 DOUB

Вектор числовых значений представляется серией компонент, конструируемых в соответствии с приведенными выше правилами, отделенных друг от друга запятыми. Например, значение ЭД с именем "X" — трехмерный вектор:

X = 1.5, 3.2, 7.6 ;

Ядро системы вмешивается при преобразовании числовых данных из внутреннего представления во внешнее и наоборот. Могут теряться знаки точности при переходе из внешнего текстового описания числа. Кроме того, количество знаков числа выводимых системой в документ регулируется инструкциями системы:

**PRECISION**,{p}

где p — число знаков, печатаемых после десятичной точки;

**SIGNIFICANCE**,{s}

полное число выводимых в документ значащих цифр числа. Параметры s и p для значений ЭД типа DOUB (действительных двойной точности) системой удваиваются.

В случае ЭД типа NODE выражение (7) преобразуется в:

{имя ЭД}; (7б)

**ВЫЗОВ процессора ввода** (IPRO = {text};)

Этот атрибут и параметры к нему описывают формат обращения системы к прикладному процессору контроля ввода значения ЭД.

Форматы атрибута IPRO используемые для согласования ядра системы и прикладного процессора:

EXTERNAL — значение ЭД подается ядром системы в IPROC и возвращается системе во внешнем представлении;

INTERNAL — значение ЭД подается системой в IPROC процессор и возвращается ей во внутреннем двоичном представлении;

CONVERT — значение ЭД подается системой во внешнем представлении и возвращается ей во внутреннем.

**ВЫЗОВ процессора вывода** (OPRO = {text};)

Описание параметров обращения системы к процессору контроля ввода значения ЭД

Допустимый формат такой же как и для атрибута IPRO за исключением того, что параметр CONVERT имеет противоположный смысл:

CONVERT — значение ЭД подается системой во внутреннем представлении и возвращается ей во внешнем.

**ВИРТУАЛЬНОЕ значение ЭД** (VIRT;)

Наличие этого атрибута значения ЭД приводит к тому, что значение ЭД не будет представлено во внутреннем представлении, и не будет документироваться из записи, однако, его можно вводить в базу данных и использовать в качестве ключевого к записи.

**КЛЮЧЕВОЕ значение ЭД** (KEY;)

Для значений ЭД с этим атрибутом система создает инвертированный список, с помощью которого производится процедура выбора записи по запросу пользователя.

Если размер описания значения ЭД системой не ограничивается и не фиксируется, то в инвертированном списке для значений ЭД выделяются поля фиксированного размера, и этот размер (LENG) является атрибутом ключевого элемента данных:

**ДЛИНА ключевой части значения ЭД** (LENG = n;)

Длина задается в машинных словах для ЭД имеющих значения любого типа. По умолчанию длина ключевой части значения равна единице.

При конструировании базы данных может возникнуть необходимость в преобразовании значения ЭД при переводе его из записи в инвертированный список. Например, если значением ЭД является фамилия и инициалы автора публикации, записанные в формате: И.О.Фамилия, то для поиска удобнее такое значение преобразовать к виду: Фамилия И.О. и в таком виде поместить в инвертированный список. С этой целью администратор базы создает прикладной процессор KPROC, на вход которому система подает значение ЭД и из которого получает его ключевую часть, если ключевой ЭД имеет атрибут:

**ВЫЗОВ процессора контроля ключевого значения** (KPRO = {text};)

Форматы атрибута KPRO используемые для согласования системы и прикладного процессора такие же, что и для IPROC.

Еще один атрибут необходим ключевому ЭД для для преобразования его значений, используемых в запросах:

**ВЫЗОВ процессора контроля запроса** (QPRO = {text};)

Для ключевых ЭД имеющих такой атрибут система обращается к прикладному процессору QPROC, согласование входа и выхода из которого определяется так же, как и для IPROC процессора.

### 1.3.2 Таблица определения структуры записи (FDT)

Описания всех ЭД и значений их атрибутов содержатся во внешнем представлении в файле FDT (File Definition Table). Запись документа FDT осуществляется инициирующей базу данных подсистемой DEFIne. Это "нулевая" запись любой BDMS базы данных. Инструкция BDMS:

**FDT**

документирует FDT запись. При использовании в ней квалификатора P:

**FDT/P**

запись FDT документируется в файл DUMP.FDT.

## 1.4 Система файлов PPDS

Структура данных BDMS внешне отображается системой файлов расклассифицированных по специализированным директориям VMS. Это файлы общего пользования: базы данных, их описания, загрузочные модули; создаваемые BDMS при работе с базой данных пользовательские рабочие файлы и файлы содержащие документы, выбранные пользователем из баз данных.

Для ознакомления с файловой структурой в PPDS-COMPAS встроена вспомогательная процедура. Обращение к ней производится через основную команду PPDS:

PPDS ?

и, после печати на экран рекламной заставки PPDS пользователю предлагается меню:

```
Select: [P]rimer, [M]anuals, [S]tructure, [R]eview of DB [U]sage > _
```

| | | |
|---|---|---|
| Primer | – | вывод на экран пользовательского дисплея структурированного описания основных правил работы с базами данных PPDS /1/; |
| Structure | – | путеводитель по всей структуре файлового хозяйства PPDS; |
| Manuals | – | путеводитель для файловой структуры MANUALS (директории, содержащие все существующие в машинном виде описания системы, баз данных, правила их сопровождения и использования); |
| Review | – | список PPDS баз данных и описания к ним; |
| Usage | – | выбор базы данных и переход к работе с нею. |
| Reviev и Manuals | – | подсистемы путеводителя Guide. |

Путеводители в некотором смысле "автопилоты" для навигации по системе файлов. Работая с ними пользователь либо переходит из директории в директорию, получая в каждой краткое описание ее содержимого, либо указывает в одной из них заинтересовавший его файл, который, в зависимости от расширения к имени может быть распечатан на экран дисплея или скопирован, по желанию пользователя в его директорию для преобразования на доступное устройство вывода.

**Все файлы, представляемые путеводителями, содержат информацию открытую для использования.**

По инструкции путеводителя ? пользователю представляется описание доступных команд:

| | |
|---|---|
| ? | печать списка инструкций; |
| < | возвращение в предыдущий контекст; |
| $ | сообщение администратору; |
| | перепечатывание списка файлов текущей директории; |
| {file} | выбор файла из предложенного путеводителем списка; |
| >{directory} | переход в директорию нижнего уровня по отношению к текущей. Имена файлов и директорий можно сокращать, если не возникает неоднозначность. |

## 1.4.1 Директории

На Рис.4. приведена схема директорий системы файлов PPDS. Жирным шрифтом выделена часть структуры PPDS поддерживающая функционирование BDMS/4, структура MAIN.

Директориям структуры присвоены логические имена, используемые в макрокомандах, например:

BDMS$COM
BDMS$MAIN
BDMS$HELP
BDMS$FDT
BDMS$COMPAS
...

### Структура MANUALS

Ознакомление с системой рекомендуется начинать с ее MANUALS части. Все созданные и большая часть создающихся описаний PPDS (описания системы в целом, описания СУБД, описания отдельных баз данных) представленны в этой структуре в виде текстовых файлов.

Структура директорий MANUALS не фиксирована, и системой BDMS не используется. Для поиска требуемых описаний необходимо прочитать имена подчиненных MANUALS директорий, которые соответствуют различным темам полного описания системы PPDS:

PPDS — обобщающие описания системы, например PPDSPRIMER /3/;

BDMS — описания СУБД, ее исходной версии и текущего состояния;

VOCABULARY — описание языка баз данных ФВЭ PPDL (Particle Physics Data Language), тезаурус PPDL.

Кроме указанных в структуру MANUALS введены все описания PPDS баз данных и правила индексирования записей этих баз. Например: директория DOCUMENTS содержит описание базы данных DOCUMENTS и описание правил индексирования: ENCODER.ENG.

Текстовые файлы в кодировке ASCII для английских текстов доступны для чтения с экрана дисплея. Среди описаний могут присутствовать описания на русском языке. Как правило, это предварительные описания, чаще всего имеющие копию на английском языке. Русские описания кодируются в формате КОИ-8 либо в семибитной кодировке с использованием "эскейп последовательностей".

Имя файла с описанием определяется его содержимым. Информация о типе описания кодируется в расширении к имени файла. Так английская версия описания будет иметь расширение "ENG", русская — "RUS". Файлы с расширением "DAT" не предназначены для вывода на экран дисплея (например VOCBOOK.DAT).

Особый статус у файлов с расширением к имени: "HLP". Это структурированное описание в котором интересующую подтему можно выбрать используя "МЕНЮ". Для вывода на экран дисплея файл обрабатывается специальной программой, с функциями аналогичными функциям встроенной в BDMS/4 подсистемы HELP. Программа включается автоматически при обращении к файлу с расширением "HLP" во время работы с путеводителем (см. выше).

Информация, хранящаяся в структуре MANUALS в основном соответствует текущему состоянию PPDS, однако более точно оно отражается самоописаниями системы.

### Структура MAIN

Структура MAIN — носитель самой СУБД BDMS/4, баз данных PPDS, системных служебных файлов. MAIN — фиксирована в PPDS дистрибутиве. Изменения в этой структуре (замена имен директорий, имен файлов) не могут быть произведены без изменения в BDMS программе, где структура MAIN задана явно.

В директории MAIN размещены все загрузочные модули, необходимые для работы с базами данных, представленными в дистрибутиве. Это файлы с общим расширением .EXE, в имени которых указан используемый данным модулем процессор контроля. Например: DOC.EXE или CS.EXE.

- MAIN — (BDMS$MAIN:) головная директория структуры MAIN. Директории подчинены:
  - BASE — (BDMS$BASE:) директория, содержащая файлы баз данных PPDS. BDMS$BASE:
  - HELP — (BDMS$HELP:) директория структурированных файлов самоописания BDMS/4 и баз данных. (см. приложение HELP). BDMS$HELP:
  - COMPAS — (BDMS$COMPAS:) вспомогательные файлы BDMS/4, содержащие внутрисистемное описание PPDL, словари орфографического контроля, файлы учета заданий, файлы инициирования заданий и автокодирования в PPDS базы данных.
- FDT — (BDMS$FDT:) Группа "FDT" файлов, определяющих структуру документа в базе данных, собрана в подчиненной директории — FDT. Эта директория

кроме FDT файлов может содержать файлы баз данных *.FIL, в которых записана только FDT информация. При наличии такого файла процедура DEFINE — определение пользовательской базы данных заменяется при использовании макрокоманды TEST простым копированием уже готового файла базы данных.

- SAVE — (BDMS$FDT:) файлы описания доступа к базам данных для их редактирования; MAIL — файлы внутренней почты PPDS.

- MAIL — структура директорий поддерживающих внутреннюю почтовую подсистему системы BDMS. Головная директория структуры содержит файл-маску STRUCTURE.DAT описывающий пользователей подсистемы.

### Структура PROC

Директория PROC является головной в структуре директорий, хранящих файлы программ и объектные модули процессоров контроля ввода/вывода в соответствующих базах данных. Структура PROC не фиксирована в PPDS дистрибутиве, т.е. изменения в ней, переименования файлов или устранение ее из дистрибутива на работе системы не скажутся.

Для каждого процессора в структуре создается директория, имеющая его имя, например: DOC - для процессора документальных баз данных, VOC — для словарной и т.д. В структуре размещаются также текст и объектный модуль ядра системы BDMS/4 и используемые ею библиотеки (Директория NUC).

### Структура TOOLS

Эта структура также не фиксирована в PPDS дистрибутиве. Головной в ней является директория TOOLS, которой подчинены директории, содержащие файлы, используемые вспомогательными процедурами, не связанными непосредственно с BDMS/4:

- TAPES — набор загрузочных модулей для процедур тестирования данных на магнитных лентах. Используется для облегчения коммуникации с группами сотрудничества по систематизации данных в ФВЭ;

- VOCBOOK — файлы с текстами программ и загрузочные модули процедуры подготовки тезауруса — справочной книги PPDL;

- SPELLER — директория файлов программ и загрузочных модулей процедур орфографического контроля и обработки используемых в нем русского и английского словарей. В версии дистрибутива за 1987 год словари содержат 50000 словоформ русского и 80000 английского языков. Орфографический контроль используется как при кодировании в базы данных текстовой информации с помощью автокодировщика BDMS/4, так и вне BDMS/4, для чего в BDMSINI.COM введен символ WORD. Обращение к процедуре в этом случае:

  WORD имя файла

### 1.4.2 Файлы

#### Командные файлы

Файлы с расширением .COM, сгруппированные в директории BDMS$COM (это и другие логические имена инициируются в BDMSINI.COM командном файле).

BDMSINI.COM

Файл содержит данные необходимые для инициирования символов и логических имен системы. При обращении к этому файлу: @BDMSINI.COM — в таблицу процесса вводятся описания команд PPDS системы: PPDS, TEST, процессоров и имен основных директорий системы отображаемых в дистрибутиве. Предполагается автоматическое использование этой макрокоманды. Параметров к ней, в том числе и запроса подсказки, нет. После работы макрокоманды инициируется внутреннее паспортное слово системы BDMS$INIT, используемое в макрокомандах PPDS, TEST и тд. Этот файл редактируется при установке системы ее администратором на новой ЭВМ. Указывается адрес головной директории системы:

XXBDMS ==”узел::имя диска:[директории...BDMS”

где BDMS — имя головной директории на схеме Рис.2.2.

PPDS.COM

Файл макрокоманды PPDS — обеспечивающей пользовательский доступ ко всем основным базам данных. Макрокоманда ”включается” по команде:

PPDS

инициируемой BDMSINI.COM. Макрокоманда снабжена короткой рекламной подсказкой:

PPDS ?

и развернутой:

PPDS ??

Макрокоманда позволяет открыть доступ к основной базе данных как в интерактивном, так и в неинтерактивном режимах. В последнем случае файл с инструкциями и данными пользователя описывается во втором параметре:

PPDS {имя базы} {имя файла}

Если при описании имени базы данных пользователем допущена ошибка, то макрокомандой предоставляется возможность уточнить имя базы данных выбрав его из списка существующих в структуре баз данных.

Макрокоманда определяет автоматически процессор, с помощью которого должна обрабатываться указанная пользователем база данных. Необходимая для подобного анализа информация размещена в файле BDMS$COM:TABLE.DAT. Базы данных не обрабатываемые процессорами в этот файл не заносятся.

TEST.COM

С помощью макрокоманды TEST создается пользовательская база данных (файлы которой размещены в пользовательской директории). Макрокоманда позволяет создать базу данных либо с уже существующей структурой записи, в этом случае будет использован соответствующий FDT файл из директории BDMS$FDT, либо со структурой сконструированной пользователем.

TEST — используется для задач пополнения основных PPDS баз данных о чем подробнее в главе 6.

Макрокоманда снабжена подсказкой:

TEST ?

### Описания

В структуру системы введены все существующие тексты описаний системы и баз данных. Эти файлы локализованы в BDMS$MANUALS:, которая разделена на поддиректории по темам.

При ознакомлении с BDMS$MANUALS с помощью путеводителя макрокомандой автоматически производится выбор операции необходимой для представления указанного пользователем файла: печать на экран структурированного описания, печать на устройство вывода. Кроме того, пользователь может заказать копию заинтересовавшего его описания в свою директорию. Структурированные описания имеют расширение .HLP.

Машинные копии оригиналов печатных статей приводятся на английском (расширение .ENG), и русском (.RUS) языках.

В поддиректории VOCBOOK размещен файл PPDL-тезауруса VOCBOOK.DAT — книгоподобная выдача представляющая собой билистинг с размером страницы 60 строк и длиной строки в 132 позиции.

### Самоописания системы

Самоописаниями здесь названы тексты, содержащие разъясняющую информацию, автоматически печатаемую системой на экран дисплея в зависимости от контекста, в котором у пользователя возникли затруднения в диалоге. Все файлы самоописаний размещены в директории BDMS$HELP:

SYSTEM.HLP

Файл содержащий описания всех инструкций BDMS. Формат его приведен в приложении HELP.

{имя базы}.HLP

Файлы имеющие формат аналогичный SYSTEM.HLP. Тексты из этих файлов выводятся на экран при интерактивной работе с BDMS/4 базой данных по инструкциями HELPBASE, FDT/?{dename}.

{имя базы}.ABS

Аннотация к базе данных. Выводится на экран дисплея по инструкции:

PPDS {имя базы} ?

### Файлы с записями баз данных

Данные во внутреннем представлении системы BDMS хранятся в трех базовых файлах прямого доступа.

| | | |
|---|---|---|
| {имя базы}.FIL | - | FDT запись и записи документов базы; |
| {имя базы}.INV | - | инвертированные списки ключевых элементов данных; |
| {имя базы}.DRC | - | файл связи между *.FIL и *.INV. |

Файлы основных PPDS баз данных размещены в директории BDMS$BASE:. Файлы пользовательских баз данных создаются и хранятся в директории пользователя.

Файл *.DRC содержит адрес начала каждой записи в файле *.FIL и длину этой записи, т.е., по два слова на каждую запись базы данных.

В инвертированном файле записаны значения ключевых ЭД вместе со списком номеров записей, в которых данное значение представлено.

Организация инвертированного файла позволяет быстро находить в списке элементов данных заказанный пользователем в запросе. Ключевые значения ЭД отсортированы (алфавитно для текстовых), и записаны в файл в многоуровневой схеме: на нижнем все значения в последовательности блоков (размер дискового блока в BDMS/4 = 512 словам для всех файлов прямого доступа), на следующем блоки содержащие списки первых из значений ЭД каждого из блоков нижнего уровня со ссылкой на этот блок, аналогично на следующем уровне, и так до тех пор, пока уровень не вместится в один блок. Подобные "деревья" создаются для каждого из ключевых ЭД и ссылки на головные блоки каждого дерева хранятся в нулевой записи *.FIL файла.

При редактировании записи как и при дополнении исправленная версия просто дописывается в конец *.FIL файла и обновляется ее адрес в файле связи *.DRC. Возникающие при этом новые или исправленные значения ключевых ЭД "вставляются" на их места в инвертированном списке. В блоках инвертированного файла запасено пространство для возможного дополнения. Размер этого пустого пространства определяется инструкцией системы:

FREE,{n}

где {n} — свободное пространство блока в процентах.

Если при пополнении свободного пространства в блоке не хватило, то переполняющийся блок делится на два (заполненых на половину каждый), и на вновь созданный из "родительского" проводится ссылка.

### FDT файлы

{имя базы}.FDT

Файлы, содержащие документы описания структуры базы данных и используемые подпроцессом BDMS DEFINE при инициировании НОВОЙ базы данных. Размещены в директории BDMS$FDT.

{имя базы}.FIL

Этот файл, размещенный в той же директории BDMS$FDT, используется при сопровождении основных баз данных. В отличии от одноименного файла базы данных, этот содержит только FDT запись и используется при создании микробаз данных.

### Вспомогательные файлы BDMS/4

Вспомогательные файлы размещаются в директории BDMS$COMPAS:. Это файлы описания модели кодирования для процедуры GUIDE ({имя базы}.GDN), файлы инициирования задания: ({имя базы}.LGF), файлы административного контроля BDMS.SPY, словари: PPDL.VOC — тезаурус используемых в PPDS базах данных, DICT.DAT —

английский словарь для орфографического контроля, RDICT.DAT - русский словарь для орфографического контроля.

### Файлы административного контроля

BDMS.SPY

В этом файле накапливается информация об использовании PPDS баз данных. Каждое обращение к базе данных отмечался в BDMS.SPY файле одной строкой содержащей описание параметров обращения: имя используемой базы данных, адрес, из которого производилось обращение и, время, потраченное системой на выполнение заказа пользователя.

{dbname}.SAV

Временный файл, размещенный в директории BDMS$SAVE. Создается в момент первой записи из системы в основную базу данных и стирается при легальном окончании работы системы с базой данных. Наличие этого файла в BDMS$SAVE означает, что произведен аварийный выход из системы и в файлах базы данных возможны ошибки. При наличии этого файла доступ к соответствующей базе данных закрыт.

Файл содержит информацию о времени старта задания приведшего к порче базы данных, описание версии загрузочного модуля, работавшего с нею и служебную информацию BDMS, которую можно получить на экране дисплея, используя при интерактивной работе с BDMS инструкцию:

@/SHOWSYSTEM

{dbname}.CNT

При наличии этого файла в директории BDMS$SAVE система открывает доступ к основным базам данных на запись только из тех директорий, которые указаны в этом файле. Санкционировать MAIN доступ к базе данных администратор может для любого количества директорий.

Формат строки этого файла:

MAIN ACCESS FROM >>> {адрес 1}

....

MAIN ACCESS FROM >>> {адрес n}

### Файлы загрузочных модулей-процессоров

{имя процессора}.EXE

Эти файлы размещены в директории BDMS$MAIN:, головной директории BDMS/4 структуры.

Файлы, содержащие тексты программ системы размещены в поддиректориях структуры BDMS$PROC и снабжены командными файлами компиляции и загрузки с помощью которых создается и размещается в BDMS$MAIN необходимый загрузочкный модуль.

Каждая поддиректория структуры BDMS$PROC по соглашению должна файл с текущей версией текста программ прикладных процессоров контроля данной базы данных, библиотечный файл, требуемый для сборки процессора, командный файл сборки и файл пояснений COMMENT.TXT, поддерживаемый администратором или программистом данного процессора.

Тексты программ ядра системы, ее объектные файлы и используемые для ее сборки файлы библиотек программ размещены в поддиректории "NUC" структуры BDMS$PROC.

Пользовательскими здесь называются файлы, размещаемые системой при работе с базой данных в директории пользователя. При общении с BDMS желательно знать о возможности появления таких файлов, их предполагаемых размерах, содержимом.

### Рабочие файлы

{dbname}.HIT

Список номеров записей попавших в выборку. Размер этого файла определяется количеством произведенных выборок и их объемом. Инструкция BDMS:

PURG

уничтожает содержимое файла *.HIT.

{dbname}.SCR

Используется для промежуточного хранения больших массивов данных.

По концу сеанса связи с базой данных рабочие файлы автоматически уничтожаются системой.

### Файлы контроля задания

{имя базы}.AUD

"АУДИТ"-файл. В нем дублируются все воспринятые за время работы с системой инструкции. При старте задания этот файл приписывается в моде дозаписи. В файле начало каждого нового задания отмечается инструкцией BDMS/4 "MESSAGE" с указанием даты старта этого задания. Файл используется в случае сбоев системы, отказов ЭВМ. Аудит файл используется для восстановления ситуации на момент нештатной остановки задания.

{имя базы}.LOG

Файл контроля, в который кроме входных данных и инструкций записывается информация обо всех отмеченных системой ошибках. Выход данных в этот файл можно перекрыть инструкцией @/NOLOG и снова разрешить: @/LOG. Кроме того, при загрузке больших объемов данных можно ограничить поток текущей в него информации используя при запуске системы параметр *NOLOG. Например:

DOC DOC *LOAD *NOLOG

### Входные файлы

Для организации неинтерактивного режима работы с системой создаются входные файлы с инструкциями системе и документами. Первая строка входного файла должна содержать одну из инструкций системы определяющую выбор подсистемы BDMS/4: LOAD, DEFINE, ADD. Инструкция ADD используется в подсистеме EXECUTIVE, которая и выбирается по умолчанию.

В общем случае файл содержащий входные данные подсоединяется инструкцией:

{процессор} {база данных} *CR{имя входного файла}

При использовании макрокоманды PPDS или TEST входной файл указывается в параметрах:

PPDS {имя базы данных} {имя входного файла}

TEST {база данных} {подсистема} {входной файл}

При работе с "большими" по объему базами данных (большими по сравнению с доступным дисковым носителем), бывает удобно использовать последовательно несколько входных файлов вместо одного. В таком случае используется инструкция:

@{имя файла}

по которой входной канал перецепляется к указанному файлу и система получает из него инструкции и данные пока файл не кончится, после чего управление вновь возвращается к исходному файлу. Например, если входные данные размещены в файлах F1.DAT, F2.DAT, ... Fn.DAT, то для чтения их системой в пределах одного фонового задания можно создать управляющий файл:

LOAD

@F1.DAT

@F2.DAT

...

@Fn.DAT

STOP

### Выходные файлы документов баз данных

Документ записи BDMS по желанию пользователя может быть размещен в одном из выходных файлов системы имеющем имя базы данных и расширение к имени, описывающее тип выходного формата представления документа. В BDMS/4 таких типов три. Это формат понятный системе (DUMP-формат). Файлы этого типа используются при экспорте и импорте данных с одной ЭВМ на другую и, по умолчанию имеют расширение к имени DUM.

{dbname}.DUM

В BDMS/4 используются два режима вывода данных в DUMP файл: редкий вывод — каждый элемент данных выводится в файл с новой строки, и, плотный вывод - документ печатается без перехода на новую строку файла на каждый элемент данных.

В файл

{dbname}.PRN

производится печать документа с описанием его структуры по инструкции PRINT.

Печать документа с использованием прикладного процессора представления данных DOCUMENT вызываемого одноименной инструкцией производится в файл с расширением DOC:

{dbname}.DOC

**Вспомогательные файлы**

Файлы содержащие модель кодирования {имя базы}.GDN, и инициирующие задание: {имя базы}.LGF пользователь может разместить в своей директории. В этом случае даже при наличии аналогичных системных файлов будет использован пользовательский.

Кроме перечисленных система может создать файлы, используемые ею для временного хранения документов. ENCODER.TMP — файл содержащий индексированные данные введенные с помощью подсистемы GUIDE, и файл создаваемый системой перед вызовом внешнего редактора (инструкции HOST, @/SPAWN): EDITOR.MRG.

### 1.4.3 Дистрибутив системы

Файловая структура PPDS в среде VMS зафиксирована в дистрибутивной версии.

**Материальный носитель дистрибутива PPDS** — магнитная лента, записанная с накопителя на диске стандартной для VMS процедурой BACKUP с сохранением структуры директорий системы. Плотность записи 6250bpi. Для удобства могут быть использованы две ленты. В таком случае первая содержит полное описание структуры директорий PPDS (120 Мб) кроме части наиболее объемных баз данных, записанных на второй ленте.

Чтение дистрибутива PPDS с МЛ на дисковый накопитель производится следующей последовательностью стандартных команд VMS:

MOUNT/FOR_"имя устройства ввода с МЛ"_"логическое имя"

BACKUP/LOG "логическое имя":PPDS [...]*.*

В результате на дисковом накопителе создается структура директорий, содержащих все необходимые файлы PPDS.

Инициирование системы PPDS после чтения с МЛ производится следующим образом:

1. В файле дистрибутива [...BDMS.COM]BDMSINI.COM редактируется строка, содержащая описание символа XXBDMS — адрес в системе VMS прочитанной версии дистрибутива, т.е. указание узла, диска, вышестоящих директорий (если они есть).

2. В пользовательские файлы LOGIN.COM (инициирующий файл VMS) нужно вставить символ:

   PPDS =="@адрес на диске...COM]PPDS.COM"

Базы данных дистрибутива PPDS. В дистрибутив 1987 введены все основные базы данных ФВЭ, за исключением базы данных INIS.

| | |
|---|---|
| **DOCUMENTS** | документальная библиографическая база экспериментальных работ в ФВЭ. Сопровождается PDG LBL (USA) (Lawrence Berkeley Laboratory) и группой КОМПАС ИФВЭ. Процессор DOC. |
| **EXPERIMENTS** | библиографическая база данных, содержащая описание текущих экспериментов в ФВЭ. Сопровождается группой PDG LBL (USA) и группой КОМПАС. Процессор DOC. |
| **REACTIONS** | фактография экспериментов в ФВЭ. Сопровождается группой КОМПАС, PDG (UK), HERA (CERN), ИТЭФ (Москва). Процессор RD. |
| **PP** | база данных по свойствам элементарных частиц. Сопровождается группой КОМПАС на основе RPP группы PDG LBL (USA). Процессор PP. |
| **CS** | компиляция сечений реакций (пополняется на основе архивов DOCUMENTS и REACTIONS) Сопровождается КОМПАС, HERA (CERN). Процессор CS. |
| **VOCABULARY** | словарная база данных PPDL. Сопровождается группой КОМПАС. Процессор VOC. |

Размер дистрибутива PPDS в основном определяется размером включенных в него баз данных. Дистрибутив-1987 = 120 Мб.

### 1.4.4 Сводная таблица PPDS файлов

В таблице приведены файлы BDMS/4 с кратким описанием данных и их функций в системе.

| ИМЯ ФАЙЛА | АДРЕС | СОДЕРЖИМОЕ |
|---|---|---|
| | | МАКРОКОМАНДЫ |
| BDMSINI.COM | BDMS$COM | инициирование PPDS |
| PPDS.COM | | доступ к основным БД |
| TEST.COM | | доступ к пользовательским БД |
| | | БАЗЫ ДАННЫХ |
| {dbname}.FIL | BDMS$BASE/USER | записи базы данных |
| {dbname}.INV | | инверт-файл |
| {dbname}.DRC | | файл связи INV-FIL |
| | | ВЫВОД из БД |
| {dbname}.AUD | USER | аудит-файл |
| {dbname}.LOG | | контроль ввода данных |
| {dbname}.DUM | | дамп-файл |
| {dbname}.DOC | | вывод документов |
| {dbname}.PRN | | структурированная печать |
| DUMP.FDT | | документ FDT |
| INDEX.DAT | | инвертированный список |
| EDITOR.MRG | | файл внешнего редактора |
| ENCODER.TMP | | файл подсистемы GUIDE |
| | | РАБОЧИЕ ФАЙЛЫ |
| {dbname}.HIT | | список записей выборки |
| {dbname}.SCR | | |
| | | ВСПОМОГАТЕЛЬНЫЕ |
| {dbname}.LGF | BDMS$COMPAS/USER | инициирование задания |
| {dbname}.GDN | | модель кодирования |
| BDMS.SPY | BDMS$COMPAS | файл контроля |
| {dbname}.SAV | BDMS$SAVE | файл контроля сбоев |
| {dbname}.CNT | | описание доступа к базе |
| PPDS.VOC | BDMS$COMPAS | контроль тезауруса |
| | | ОПИСАНИЯ |
| {dbname}.HLP | BDMS$HELP | файл самоописаний базы |
| {dbname}.ABS | | |
| SYSTEM.HLP | | самоописание системы |
| VOCBOOK.DAT | BDMS$VOCBOOK | тезаурус PPDL |
| | | ПРОГРАММЫ |
| {proc.}.EXE | BDMS$MAIN | загрузочный модуль |
| BDMS.EXE | | беспроцессорный BDMS |
| BDMSNUC.FOR | BDMS$NUC | текст программы ядра BDMS |
| BDMSNUC.OBJ | | объектный файл ядра BDMS |
| {proc.}.FOR | BDMS$PROC | тексты процессоров |

Рис. 1.2: На верхнем уровне структуры — основные директории: COM — директория, содержащая системные макрокоманды; MANUALS — головная директория структуры, содержащей описание системы PPDS; TOOLS — вспомогательные процедуры PPDS, не связанные с BDMS/4; MAIN — головная директория структуры файлового хозяйства BDMS/4; PROC — головная директория структуры, содержащей введенные в PPDS процессоры контроля данных.

# Глава 2

# Управление данными

## 2.1 Инициирование базы данных

Преобразование данных из записи в документ и обратно осуществляется в зависимости от значений атрибутов элементов данных, совокупность которых, FDT, можно считать оператором такого преобразования.

<запись> = FDT <документ>

<документ> = FDT <запись>

Ввод документа в существующую базу данных возможен только если структура записи в ней уже определена в ее "нулевой" записи — FDT.

На Рис.2.1 схематически отображен процесс ввода данных в БД. Система считывает FDT запись из БД и, на основании ее производит анализ входных данных из документа, их структурирование, перевод во внутренний формат. Вывод данных осуществляется также с помощью FDT.

Определяющая FDT запись БД по структуре отличается от остальных записей БД. FDT записи всех баз данных можно считать записями ВИРТУАЛЬНОЙ БАЗЫ ОПИСАНИЙ БАЗ ДАННЫХ. Физически, в виде последовательности записей она не существует, ее записи "разбросаны" по определяемым ими базам данных. Однако, основной атрибут BDMS базы данных у виртуальной БД есть: собственная FDT запись, определяющий преобразование FDT-документа в FDT-запись:

<FDT-документ —main FDT— FDT запись>

На Рис.2.1а Основной FDT встроен в систему. В BDMS/4 он не изменяем.

Структура записи заинициированной и уже заполняемой BDMS базы данных фиксирована, то есть FDT-запись такой базы модифицировать нельзя.

<FDT запись — системный FDT — FDT документ>

Как при инициировании базы данных, так и при вводе данных в уже созданную базу система использует одну и ту же подсистему: редактор (EDIT). Описание этой процедуры см. в следующей главе.

Инициирование BDMS базы данных, производится подсистемой DEFINE, обращение к которой при работе в интерактивном режиме:

TEST {имя новой базы} DEFI

в неинтерактивном:

TEST {имя новой базы} *CR{file}

где file — имя файла, содержащего инструкции системе и данные FDT-документа. ПЕРВАЯ строка этого файла должна содержать инструкцию BDMS:

DEFI

Пусть имя "новой базы данных": DOC. В интерактивном режиме система приглашает пользователя к общению титулом:

DOC:READ:DEFI> _

Возможные инструкции:

COMPILE вызов процессора контроля FDT данных;

DEFINE инициирование процедуры описания новой базы;

GUIDE вызов процедуры "автокодировщик";

HELP вспомогательное описание процедуры DEFINE;

LINE описание максимального размера входной строки;

MODIFY вызов редактирующей процедуры EDIT;

STOP выход из системы.

Входные данные — FDT-документ — записаны в виде последовательности элементов данных в формате:

{имя}={значение};

которая, в общем случае, может перемежаться инструкциями системе. Система преобразует их в запись инициируемой базы данных используя основную FDT-запись:

| имя ЭД | тип значения | родитель | |
|---|---|---|---|
| FILE | CHAR | | имя базы данных |
| DE | CHAR | | имя элемента данных |
| SYN | CHAR | DE | синоним имени ЭД |
| TY | CHAR | DE | тип значения |
| VIRT | NODE | DE | виртуальный ЭД |
| KEY | NODE | DE | ключевой ЭД |
| LENG | INTE | KEY | длина ключевого ЭД |
| IPRO | CHAR | DE | процессор IPROC |
| OPRO | CHAR | DE | процессор OPROC |
| KPRO | CHAR | DE | процессор KPROC |
| QPRO | CHAR | DE | процессор QPROC |

Несколько более наглядно схематическое изображение этой записи, которая приведена на рис.2.2 Такая форма описания FDT часто используется во вспомогательных описаниях баз данных.

Инструкция @/FDT подсистемы DEFI распечатает документ этой записи на экране дисплея, вид которой представлен в таблице 1.

Имена ЭД системного FDT, как видим, это наименования приведенных выше атрибутов элементов данных определяемой базы данных. DE — имя ЭД, SYN — его синоним, TY — тип значения и т.д. В данном FDT только один ЭД уровня записи — DE. Конструирование базы данных — это создание документа FDT в котором каждый ЭД определенный по формату 7) является описанием значения одного из атрибутов одного из ЭД создаваемой базы данных.

Ввод и анализ документа с помощью системного FDT закончится по инструкции редактору: "**". Следующей инструкцией при инициировании БД могут быть либо MODIFY — повторное обращение к редактору для внесения изменений в только что введенный документ, либо вызов системного процессора контроля данных FDT-документа: COMPILER. При отсутствии формальных ошибок в "LOG" файл системой будет передано сообщение: "COMPILATION SUCCESSFUL", сформируется запись FDT и будет помещена в начало файла TEST.FIL.

Выход из подсистемы DEFI осуществляется по инструкции STOP, (или ˆ Z). Теперь база данных создана и можно приступить к ее сопровождению (Рис.2.1б): заполнению документами соответствующими FDT, хранению, модифицированию, поиску.

## 2.2 Пример базы данных

Приведем для примера документ FDT, описывающий документальную базу данных аналогичную PPDS базе данных рефератов экспериментальных работ в ФВЭ "DOCUMENTS".

Ограничим индексируемую часть предметной области ФВЭ для такой базы данных библиографическим описанием источника информации и кратким фактографическим описанием содержимого: реакция, энергия взаимодействия, участвующие во взаимодействии частицы.

Первым элементом данных назначим "КОД ИСТОЧНИКА", значение которого должно определяться значением других элементов данных документа этой БД, быть уникальным в пределах БД.

```
DE = SC; SYN = SHORTCODE; TY = CHAR; KEY; LENG = 4;
IPRO = EXTERNAL; KPRO = INTERNAL;
```

означает, что определяется ЭД "SC", имеющий синоним "SHORTCODE", текстовый тип значения, уровня записи (не указан родитель), ключевой с длиной ключевой части значения равной 4 машинным словам, или 20 символам (байтам). ЭД предполагается контролировать процессорами IPROC и KPROC, и, кроме того, дополнительный контроль этого ЭД будет проведен SPROC процессором (см. ПРОЦЕССОРЫ BDMS).

Продолжим документ:

```
DE. = R; SYN = REFERENCE; TY = CHAR; KEY; LENG = 5;
KPRO = INTERNAL;
DE. = TY; PAR = R; TY = CHAR; KEY; LENG=1;
IPRO=EXTERNAL;
DE. = D; PAR = R; TY = CHAR; KEY; LENG=1;
IPRO=EXTERNAL;
DE. = AUTHORS; SYN = AUTH; TY = NODE;
DE. = A; PAR = AUTHORS; TY = CHAR; KEY; LENG = 4;
KPRO = EXTERNAL;
DE. = SPOKESMAN; SYN = SPOK; PAR = A; TY = NODE; KEY;
```

```
DE. = I; PAR = AUTHORS; TY = CHAR; KEY; LENG = 1;
DE. = T; SYN = TITLE; TY = CHAR;
DE. = REAC-DATA; TY = NODE; KEY;
DE. = RE;TY=CHAR;PAR = REAC-DATA; KEY: LENG = 40;
DE. = PLAB;PAR = REAC-DATA; TY=REAL; KEY; LENG=2;
IPRO=CONVERT; OPRO=CONVERT;KPRO=INTERNAL;
QPRO=CONVERT;
DE. = FSP; TY=INTE;VIRT;KEY;LENG=1;QPRO=CONVERT;
DE. = PART-PROP; TY = NODE; KEY;
DE. = P;PAR = PART-PROP; TY=CHAR; KEY; LENG=1;
IPRO=CONVERT; OPRO=INTERNAL; KPRO=INTERNAL;
QPRO=CONVERT;
DE. = PP;PAR = PART-PROP; TY=CHAR; KEY; LENG=1;
IPRO=EXTERNAL; KPRO=INTERNAL;
DE. = DEN;TY=CHAR; KEY: LENG = 2;
DE. = DCH;TY=CHAR; KEY: LENG = 2;
```

В одном документе несколько раз использован ЭД уровня записи "DE". Чтобы не сработал механизм автоматического создания родительского элемента, т.е. новой записи (см. описание процедуры EDIT), к имени повторяемого ЭД необходимо прибавить символ "." Другие имена ЭД: "TY", "SYN", "KEY" мы в примере тоже используем многократно, но все они подчинены родительскому ЭД "DE", как видно из описания системного FDT, и точку к их имени можно не добавлять.

С помощью подсистемы DEFINE описанной выше, создадим пустую базу данных преобразовав FDT документ в FDT запись базы данных.

В результате действия подсистемы DEFINE мы получим один из базовых файлов DOC.FIL В версии BDMS/4 к пустой базе данных, то-есть, имеющей FDT запись, можно обратиться только через подсистему LOAD (загрузчик), введя в базу данных один или более документов. Подробнее процедуру заполнения базы данных мы рассматриваем ниже, здесь еще несколько слов о ее "нулевой" записи. Аналогично тому, как мы получили документ основной FDT записи системы, используя ту же инструкцию FDT мы можем ознакомиться с FDT записью DOC базы данных, приведенной в таблице 2.

Ее схематичное изображение представлено на рис.2.3.

Для пользователя, знакомого со структурой данных BDMS, документ FDT может оказаться вполне достаточным справочным пособием при работе с базой данных. По этой причине инструкция, обеспечивающая доступ к документу FDT всегда "под рукой". В любом контексте при работе с системой, документ FDT печатается по инструкции общего вспомогательного уровня (см. главу 3.):

@/FDT

## 2.3 Заполнение базы данных

В BDMS предусмотрено несколько способов заполнения данных. Самый быстрый осуществляется процедурой LOAD. Эта процедура используется для загрузки инициированной, т.е. имеющей FDT запись, но не имеющей других записей базы данных. Используется при создании новых БД или перезагрузках старых. В BDMS/4 подсистема LOAD работает только в неинтерактивном режиме. Обращение к этой подсистеме осуществляется под VMS инструкцией:

TEST DOC LOAD {имя файла}

Входной файл содержит инструкции подсистемы LOAD и документы базы данных. Первая строка входного файла должна среди прочих, обязательно содержать инструкцию LOAD.

Инструкции подсистемы:

- LOAD вызов РЕДАКТОРА системы для чтения данных;
- FREE,{n} уровень заполнения блоков инвертированного файла;
- LINE описание максимального размера входной строки;
- STOP выход из системы.

В подсистеме LOAD документы последовательно читаются РЕДАКТОРОМ, записываются в файле *.FIL запись за записью, а значения ключевых ЭД выписываются вместе с номерами породивших их записей в рабочий файл *.LIN. По окончании ввода данных значения ключевых ЭД сортируются по частям методом быстрой сортировки с использованием второго рабочего файла после чего одинаковые значения объединяются так что, возникает упорядоченный список значений ключевых ЭД, каждому из которых приписан упорядоченный список номеров записей из которых взято данное значение. Окончательно инвертированные списки по ключевым ЭД переписываются в файл *.INV, после чего база данных готова к использованию (хранению пополнению и поиску).

Существует иной способ заполнения базы данных документами. Инструкция подсистемы EXECUTIVE ADD:

ADD,{n}

где n — REC-ID значение для вводимого документа, если оно не занято уже существующим в БД.

Ввод документа с помощью этой инструкции несколько медленнее чем при использовании подсистемы LOAD, (в зависимости от структуры записи и от количества ключевых ЭД в ней в три — семь раз).

Как и DEFINE инструкция рассмотренная выше, ADD передает управление системному редактору EDIT, в котором на основе данных FDT записи анализируется вводимый документ (Рис 3.6). FDT запись автоматически считывается в память при старте системы.

Приведем для примера документ нашей базы данных DOCUMENTS соответствующей препринту (TY=PREP;) группы авторов, опубликованному так же как статья (TY=JOUR;) в журнале Physical Review Letters 33, стр. 1404 (R=PRL 33, 1404;) за 1974 (D=1974;).

По фамилии первого автора (A=J.J.Aubert;) и году публикации сформирован краткий код:

(SC=AUBERT 74D;), где символ "D", четвертый в латинском алфавите, означает, что в базе данных есть и предыдущие три работы этого автора за 1974 год.

Авторы статьи приведены двумя группами. 13 авторов из института MIT, (Massachusetts Institute of Technology, Cambridge, MA. USA) и один из BNL (Brookhaven National Laboratory, Upton, NY. USA).

И статья и предшествующий ей препринт называются одинаково, что отражено в ЭД (T=Experimental observation of a heavy particle J;).

Эксперимент, описание результатов которого приведено в статье, проводился на ускорителе лаборатории BNL, использовался детектор DAS (Double Arm Spectrometer). Изучалась реакция взаимодействия протонов на ядрах берилия

```
(RE = P BE --$>$  E-  E+  X;  P  BE  --$>$  J/PSI(3097)  X;)
```

при 20 GeV/c (PLAB=19.0,20.0;) с образованием электрон-позитронных пар, в спектре масс которых исследовалось узкое состояние с массой 3.097GeV, определенное как новая частица J/PSI(3097); (PP=EX; MASS;)

Формат приведенного ниже документа определен языком описания данных, подробное рассмотрение которого в следующей главе.

```
SC = AUBERT 74D;
R = PRL 33, 1404;
TY = JOUR;
D = 1974;
R. = COO-3069-271;
TY = PREP;
D = 1974;
AUTHORS;
A = J.J.Aubert;
A. = U.Becker;
A. = P.J.Biggs;
A. = J.Burger;
A. = M.Chen;
A. = G.Everhart;
A. = P.Goldhagen;
A. = J.Leong;
A. = T.McCorriston;
A. = T.G.Rhoades;
A. = M.Rohde;
A. = S.C.C.Ting;
A. = Sau Lan Wu;
I = MIT;
AUTHORS.;
A = Y.Y.Lee;
I = BNL;
T = Experimental observation of a heavy particle J;
```

```
AC = BNL;
DE = DAS;
REAC-DATA;
RE = P BE --$>$ E- E+ X;
RE. = P BE --$>$ J/PSI(3097) X;
PLAB = 19.000,20.000;
PART-PROP;
P = J/PSI(3097);
PP = EX;
PP. = MASS;
DEN = 76/06/12  10.31.21;
DCH = 78/06/11  09.48.23;
**
```

В примере приведен документ составленный из элементов данных, атрибуты которых мы ввели ранее при инициировании базы данных. В конкретном документе любой элемент данных можно опустить, но нельзя использовать элементы данных, описания которых в FDT отсутствуют. В этом случае все описание нелегального ЭД будет воспринято как очередное значение ранее введенного легального ЭД. Такая ситуация возможна, и возникает в практике сопровождения PPDS баз данных при ошибках в имени элемента данных (см. 3.1. Множественная генерация элементов данных).

## 2.4 Поиск. Создание выборки записей

В первой главе мы уже рассматривали простейший вид поиска записи в базе данных на примере выбора по значению системного элемента данных REC-ID;. При поиске записей по ключевым ЭД используются та же инструкция FIND, но поиск ведется по индексу — упорядоченному списку ключевых значений указанного элемента данных. Упорядоченный список (ИНВЕРТИРОВАННЫЙ), хранится в отдельном файле прямого доступа: {имя базы}.INV в формате, позволяющем находить заказанное значение КЭД или убеждаться в его отсутствии при минимальном количестве чтений из файла.

Блок в INV-файл BDMS фиксированного размера. В нем содержатся значения КЭД и к каждому список номеров записей, в которых это значение представлено. При обнаружении требуемого значения BDMS переписывает все соответствующие ему номера записей в список отобранных записей: выборку. Если условию запроса удовлетворяют несколько значений, все содержащие их номера записей заносятся в выборку. При этом один и тот же номер может быть представлен в выборке несколько раз. (Например, приведенный выше документ, при отборе по авторам, фамилия которых начинается на "B" попадет в выборку трижды). Чтобы избавиться от проблем, возникающих при многократном повторении в выборке одного и того же номера записи, выборка системой упорядочивается по номерам записей (REC-ID), и из получившегося списка удаляются дубли.

При использовании сложных запросов система производит каждый раз сравнение уже упорядоченных выборок.

Выборка в BDMS идентифицируется по номеру. В запросе, язык описания которого подробно рассматривается в следующем разделе, номера выборок и описания ключевого элемента данных — равноправны.

Для хранения выборок во время работы системы используется рабочий файл {имя базы}.HIT. Этот файл создается в пользовательской директории и, по концу работы системы уничтожается. Поскольку в выбоке представлены только номера отобранных записей, то рабочий файл не превышает разумных пределов (максимум 300 – 400) блоков для больших баз данных при отборе всех ее записей. Однако, создавая много выборок (в системе на их количество нет ограничения), необходимо помнить о существовании этого файла и не допускать превышения им размеров выделенной пользователю директории. В системе для этой цели используется инструкция PURG — позволяющая либо полностью очистить список выборок, либо устранить из него последнюю.

В зависимости от запроса на процедуру создания выборки может быть потрачено от долей секунды до нескольких минут процессоного времени. Быстро производится выбор по номеру записи, при явно заданном значении КЭД; медленнее при указании в запросе области удовлетворяющих его значений; еще медленнее — поиск по содержимому значения КЭД, при котором не может быть использована упорядоченность списка значений. Дополнительное замедление в процедуру поиска вносят прикладные процессоры. Усредненная оценка на основе данных файла административного контроля использования PPDS баз данных дает около 30 сек на выбор и вывод данных на одно пользовательское обращение к системе. Однако, наличие "долгоиграющих" запросов делает заметным один из недостатков BDMS/4 — отсутствие возможности сохранить созданные выборки по базе данных, воспользоваться ими при следующем общении с этой же базой данных. Работа над системой в этом направлении ведется, и подобная возможность должна будет реализована в следующей версии (см. гл. 9).

## 2.5 Язык запросов

Язык запросов позволяет пользователю находить в базе данных записи, удовлетворяющие любому сложному условию на значения ключевых ЭД. Запрос является конструкцией из описаний условий налагаемых на значения ЭД связанных булевыми операторами NOT, AND, OR. Возможно также производить поиск по условию просто существования данного ЭД не учитывая его значение или искать генерации вовсе не имеющие значений. Для ключевых значений ЭД текстового типа можно использовать поиск по усеченному с конца значению, т.е., только по началу текста значения, или по совпадению текстовой подцепочки символов в значении ЭД и в запросе, "контекстный поиск" (2.5.5).

Подсистема QUERY, обрабатывающая запрос, включается инструкцией FIND подсистемы EXECUTIVE. В инструкции системе команда FIND предшествует запросу, описание которого заканчивается "**".

FIND {запрос} **

(Пробел перед "**" не обязателен).

Выборке записей удовлетворивщих запросу присваивается идентификационный номер, номер выборки. При этом печатается сообщение о числе записей в выборке.

{n} records in set {s}

Номер выборки {s} может быть использован в запросе в комбинации с другими условиями.

{n} может быть равным нулю. В этом случае тоже создается выборка, которая на равных может участвовать в дальнейших описаниях запросов, что полезно при работе в неинтерактивном режиме.

## 2.5.1 Простое условие

Формат простого условия:

```
                    {[= ]            [  [[=]]          ]}
                    {[> ] <знач.1>; [TO[ < ] <знач.2>;]}
                    {[>=]            [  [<= ]          ]}
<имя ключевого ЭД>  {[< ]                               }
                    {[<=] <значение>;                   }
                    {[<>]                               }
                    { ;                                 }
```

где скобки "{ }" обрамляют набор допустимых вариантов, один из которых должен быть использован, а "[]" — необязательные элементы. Операторы отношения $<=$, $>=$, $<>$ означают **"меньше или равно"**, **"больше или равно"** и **"не равно"**, соответственно. Таким образом можно специфицировать точное указание верхней и нижней границ для области искомых значений.

Примеры допустимых описаний условий поиска:

```
A = Ting;
A <> Smith;
PLAB <= 7;
PLAB = 5; TO 8;
PLAB > 5; TO < 8;
```

Последние два примера различаются тем, что во втором в область поиска включены граничные точки.

Условие

{имя ключевого ЭД} $<>$ {значение};

удовлетворяет записям в которых:

а) имеется как минимум одна генерация указанного ЭД, и

б) не имеется генераций этого ЭД значение которых равно указанному.

Если в простом условии после оператора отношения не приведено значение, то производится поиск записей в которых в генерациях этого ЭД значение отсутствует.

A =;

приведет к отбору всех записей, имеющих пустое значение ЭД "A", а

A $<>$;

определит дополняющую выборку по этому же ЭД.

И наконец, формат простого условия:

{имя ключевого ЭД};

определит выборку всех записей имеющих генерации указанного ЭД вне зависимости от их значений.

Формат описания значения зависит от типа ЭД и такой же, какой принят при редактировании (см. гл. 3). Ключевые значения в индексе имеют фиксированную длину и получены из соответствующих значений ЭД усечением или дополнением их пробелами до фиксированного размера. При обработке системой запроса будут таким же образом усечены или расширены значения КЭД формирующие запрос, что гарантирует сравнение с индексным значением.

### 2.5.2 Составное условие

В общем случае описание запроса, воспринимаемое поисковой подсистемой, конструируется из простых условий и ранее полу- ченных номеров выборок в соответствии со следующим рекурсив- ным соотношением:

```
                      [[простое условие]] [ [AND] <условие>]
условие  = [NOT]
                      [[номер выборки]  ] [ [OR ]          ]
```

т.е. простые условие и ранее определенные выборки, идентифицируемые номерами, могут комбинироваться с использованием булевых операторов NOT, AND и OR. NOT имеет высший приоритет а OR низший, но этот порядок можно изменять используя скобки "()".

Отметим, что простое условие и номер выборки играют одинаковую роль в составном условии. Можно считать, что каждое простое условие можно рассматривать как описание промежуточной выборки. Эти выборки затем комбинируются: объединяются (OR), пересекаются (AND) и дополняются (NOT).

Два следующих условия:

{имя ключевого ЭД} <> {значение};

и

NOT {имя ключевого ЭД} = {значение};

не эквивалентны. Смысл первого объяснен выше, второе отличается от первого тем, что ему удовлетворяют также записи не имеющие генераций соответствующего ЭД. Т.е. эффект действия оператора NOT не может быть получен при использовании только операторов отношения:

```
NOT  =                              <>
NOT  <                              >=
NOT  >      не эквивалентно         <=
NOT  <=                              >
NOT  >=                              <
NOT  <>                              =
```

### 2.5.3 Идентификатор записи (REC-ID).

Каждой записи в базе данных присваивается уникальный и постоянный целочисленный идентификатор при ее заполнении в базу данных. ЭД REC-ID может быть использован в простом условии запроса для поиска записей по их идентификационным номерам. Например:

FIND REC-ID;

приведет к образованию выборки, содержащей все записи базы данных.

### 2.5.4 Поиск по усеченному значению

Описание усеченного значения в запросе позволяет искать записи содержащие генерации указанного ЭД чьи значения начинаются указанным текстом, а остальной текст по условию поиска безразличен. Например, можно захотеть найти все статьи написанные любым Смитом, вне зависимости от его имени, если автор описывается сначала фамилией. Усечение значения отмечается символом "/" следующим за частью значения. Т.е. описанный пример в запросе выглядит:

FIND A=Smith/; **

все записи, имеющие значения ЭД "A" чьи первяе пять символов "Smith" будут удовлетворять этому условию.

Если в описание значения в запросе необходимо вставить символ "/" (как составную часть значения) то он печатается дважды: "//".

Усеченное описание значения можно использовать в сочетании с любым оператором отношения. Любая или обе граничные точки области поиска могут быть описаны усеченными значениями. При этом нужно помнить, что соотношение "=" определяет не одно, а сразу некоторую область значений.

### 2.5.5 Поиск по содержимому значения ключевого ЭД

В случае текстовых значений ключевых элементов данных возможность выбора только по полному или усеченному с конца значению КЭД может оказаться недостаточной или недостаточно удобной. Возможности выбора расширяются при использовании анализа содержимого значения КЭД. Выборка по содержимому производится, если при имени КЭД в стандартном простейшем запросе указывается квалификатор CONTAIN:

FIND <имя КЭД>/CONTAIN = условие выбора; ...

Простейшее описание условия выбора — контрольная цепочка символов совпадение которой с любой подцепочкой в значении КЭД является критерием выбора.

Символ "%" в контрольной цепочке отмечает положение "любого" символа. Например:

AGINENKO

AJINENKO

удовлетворяют условию выбора:

A%INENKO

Символом "*" отмечается положение "любой" подцепочки символов внутри контрольной цепочки символов. В понятие "любой" здесь включено: любая длина цепочки из любых символов. Например:

AZHINENKO

AJINENKO

AGINENKO

удовлетворяют условию выбора:

A*INENKO

Можно указать номера позиций символов в значении КЭД по которым, и только по ним должно осуществляться сравнение с условием выбора: зону контроля. С этой целью в условие выбора вводится следующее описание:

$: n_1 \ - n_2 \ - n_3 \ - n_4 \ldots n_k \ - n_{k+1}$ ;

где $n$ — номер позиции в цепочке символов КЭД. $n_1 \leq n_2 \leq \ \ldots$ — условие, при соблюдении которого приведенная конструкция рассматривается как описание зон контроля.

На рис.2.4а закрашены зоны контроля соответствующие указанным в [1]. При сравнении с условием выбора [2] представляется процедуре контроля как непрерывная цепочка символов показанная на рис.2.4б.

Если первый номер в [1] записан с минусом:

$$: -n_1 \ldots . :$$

то им отмечается зона контроля от начала цепочки до позиции $n_1$, как показано на рис.2.4в.

Аналогично, если последний в конструкции номер не имеет "пары", то зоной контроля будет от $n_{k+1}$ до последнего символа значения КЭД.

В контрольной цепочке символов кроме "%" — "любого" можно ввести описание символа "одного из ..." При этом существенно используется порядок символов в ASCII кодировке.

Конструкция:

>A<Z

означает, что стоящий на ее месте в контрольной цепочке символ — одна из букв от A до Z, но не цифра и не какой-либо другой символ. Совмещая подобные конструкции можно указать допустимые "зоны" для возможных символов:

$$> s < s_2 > s_3 < s_4 \ldots .$$

где s — символ. Как и для описания [1] требуется выполняемость условия:

$$s_1 \leq s_2 \leq s_3 \ldots \leq \ s_n .$$

Это описание соответствует при этом только ОДНОМУ символу. "Зоны" для символов описываются по тем же правилам, что и "зоны контроля". Например:

<A

означает: "все не буквы".

В условие выбора можно указать цепочку символов, наличие которой в значении КЭД запрещает его выбор, условие "NOT". Формат такого описания:

#текст

Например, условие:

A#B

выполняется для тех значений, в цепочке символов которых есть хотя бы один символ "A" но нет ни одного символа "B".

В одном условии выбора можно с помощью логических операторов из простых условий сконструировать сложное. Логические операторы обозначаются символами: "$" — AND, и "&" — OR.

Пример такой конструкции:

ABC$DEF$1A

при которой критерием выбора является наличие в цепочке символов значения КЭД подцепочек "ABC" и "DEF" и "1A".

При описании условия выбора символы "*", "#", "$", "&" использованы в качестве команд языка описания. Если необходимо, чтобы эти символы участвовали в сравнении, их нужно печатать дважды. Например:

A**B>>C

описывает условие выбора

A*B>C в котором уже ни "*" ни ">" не являются метками языка описания.

Кроме инструкции FIND процедура выбора значения КЭД по анализу его содержимого работает в инструкции INDEX/CONTAIN=...; (см. 5.1).

При формировании сложного запроса квалификатор /CONTAIN действует только на выбор по тому КЭД, при котором он указан:

FIND A/CON = AZH; AND I = SERP; **

В этом примере заказана выборка документов, содержащих ЭД "I" со значением "SERP", и ЭД "A" в значении которого встречается цепочка символов "AZH".

ПРИМЕЧАНИЕ: процедура поиска по содержимому может быть использована и при выборе по значениям КЭД числовых типов. При этом данные о формате этих значений преобразованных для выбора в текстовое представление можно получить с помощью инструкции INDEX.

## 2.6 Манипулирование выборками записей

Любое обращение к записи базы данных может производится в BDMS только через предварительно созданную выборку, содержащую эту запись. В выборке отдельная запись "приобретает" еще один идентифицирующий номер: порядковый номер записи в выборке. Именно по нему в дальнейшей работе с выборкой идентифицируется конкретная запись.

Вспомним еще раз: системный идентификатор REC-ID — это номер записи в базе данных. Доступ к записи по этому номеру по инструкциям GET,{n} или FIND REC-ID...;** После того, как выборка создана, доступ к записи в ней осуществляется по ее порядковому номеру в выборке, обозначаемой в тексте {nrec}. Обозначение NREC будем использовать для полного числа записей в выборке.

Все манипуляции с записями: редактирование, документирование, удаление их из базы данных, сортировка производятся после того, как необходимые записи отобраны в выборку.

### 2.6.1 Документирование записей

В BDMS различаем следующие формы документа: "DUMP" — экспорт/импортный вариант представления; "LIST" — документирование на экран дисплея в формате, подобном "DUMP" формату, но визуализуется структура записи; "PRINT" - то же, что "LIST" но документирование в файл имя базы.PRN; и специальный формат, определяемый прикладным процессором вывода. Если в формах "DUMP", "LIST", "PRINT" при документировании в общем сохраняется не только деление информации в базе данных на записи но и отражается структура записи, то формат "DOCUMENT" уже необратим. В зависимости от задачи процессор вывода DOCUMENT может во внешнем представлении объединять различные типы данных из разных записей представляя их совместно.

Перечисленные формы документирования записей могут быть дополнены внешним преобразованием одной из форм документа специальными процедурами. Так например, процедура CSBOOK — преобразует документы базы данных CS (cross sections) в книгоподобный файл, аналог которого - широко известные публикации ЦЕРН'овской группы HERA. Упоминавшийся в гл. 1. книгоподобный файл VOCBOOK.DAT получается преобразованием VOC.DUM документов специальной процедурой VOCPREP.

Документирование записи производится одной из инструкций подсистемы EXECUTIVE (см. гл. 5): LIST, DUMP, PRINT, DOCUMENT. Формат первых трех одинаков: указываются номера {nrec} записей для документирования и следом список документируемых ЭД завершаемый точкой. По умолчанию документируются все записи текущей выборки если не заданы номера и все ЭД, если не указан список. Для удобства пользования документирующими инструкциями введены дополнительные инструкции SUPPRESS — определяет список по умолчанию не документируемых ЭД и, DISPLAY — вычеркивает указанные ЭД из списка заданного инструкцией SUPPRESS.

Инструкция DOCUMENT не имеет жестко заданного формата. Ее использованию должно предшествовать ознакомление с форматом вывода документов конкретной базы данных: инструкция HELPBASE гл. 5, или @/HELPBASE гл.4.

### 2.6.2 Удаление записей

Запись из базы данных (БАЗЫ ДАННЫХ!) удаляется инструкцией:

DELETE,{список nrec текущей выборки}

Не существует /пока/ столь же простого способа удаления записи из выборки. Поскольку такая необходимость может возникать, то рекомендуемый способ - создать отдельно вспомогательную выборку с записями, которые необходимо удалить из текущей и объединить их инструкцией FIND с использованием оператора "NOT".

Дважды выделенное в первом предложении этого раздела указание на объект действия инструкции DELETE — предупреждение пользователям собственной базы данных. При работе с основными PPDS базами данных в моде доступа READ инструкция

DELETE производит временное действие и для базы данных не опасна.

Записи, удаленные из базы данных инструкцией DELETE продолжают числится во всех ранее созданных выборках, в которые были отобраны. Более того, инструкция FIND — оперирующая над выборками, например:

FIND (1) AND (3)**

не чувствительна к опустошениям, произведенным в выборках инструкцией DELETE.

Прямое обращение к записи, уже удаленной из базы данных, но представленной в выборке приведет к предупреждению на экране:

"FETCH ERROR" Deleted record.

Похожая проблема возникает и в том случае, если в базу данных после создания выборки была введена запись, удовлетворяющая условиям формирования выборки. На этот раз запись в выборке представлена не будет, пока выборку не перенабрать, (что охватывает и предыдущий случай с удалением записи).

### 2.6.3 Редактирование записей

Изменение в записи может быть произведено двумя различными путями. Во-первых, запись может модифицироваться "напрямую", в виде записи; во-вторых, запись можно документировать, переведя во внешнее текстовое представление, отредактировать и снова загрузить в базу данных.

Первый путь реализуется инструкцией подсистемы EXECUTIVE:

MODIFY[,nrec]

Если выборка содержит только одну запись, то ее {nrec} можно не указывать.

Эта инструкция передает управление подсистеме РЕДАКТОР, описанной в гл. 3. В РЕДАКТОРЕ производится редактирование непосредственно записи. Это позволяет при редактировании не "зацепить" случайно структурные ссылки, связывающие между собою ЭД. Однако, как и любой редактор, этот требует некоторых временных затрат на его освоение. РЕДАКТОР BDMS пока лишен экранных возможностей. Поэтому, предлагается альтернативный путь модифицирования записей базы данных:

EDITOR,{nrec}[,VMS имя редактора][,параметры редакции]

Инструкция документирует указанный (или указанные) nrec записи в рабочем файле EDITOR.MRG в формате DUMP. По концу документирования управление из BDMS передается указанному редактору (по умолчанию имя редактора CONFED. Что за объект скрывается за этим именем определяется в соответствующей строке инициирующего файла BDMSINI.COM (гл. 1.)). При передаче управления BDMS формирует командную строку:

CONFED EDITOR.MRG

Результат редакции по завершении процедуры CONFED должен размещаться в файле

EDITOR.MRG (это условие удовлетворяется для большинства текстовых редакторов).

По концу редактирования управление вновь возвращается системе BDMS, которая загружает отредактированный документ в базу данных.

Более подробно эта инструкция описана в гл 5.

### 2.6.4 Сортировка записей

Записи в выборке упорядочены по их REC-ID номерам в порядке возрастания. Упорядоченность записей: внутрисистемное требование: удобнее производить процедуры объединения различных выборок.

Если возникает необходимость переупорядочить записи в выборке перед их документированием по значению какого-либо другого ЭД, то можно использовать инструкцию:

SORTOUTPUT,{имя ЭД1}[,имя ЭД2][,...]{.}

В результате будет создана выборка записей упорядоченных по значениям указанных в списке ЭД. В процедуре группового усреднения ЭД и КЭД используются на равных основаниях.

Рис. 2.1: а). Инициирование БД. (DEFI). б). Сопровождение базы данных. (EXEC, LOAD, DUMP).

Рис. 2.2:

**Таблица 1.**

| | |
|---|---|
| File Definition | Table for the base TEST |
| | Last records number: 0 |
| | Number of the data elements: 12 Key DE's 0 |
| C | ... FILE |
| C | ... DE |
| C | ... SYN |
| C | ... PAR |
| C | ... TY |
| N | ... VIRT |
| N | ... KEY |
| I | ... LENG |
| C | ... IPRO |
| C | ... OPRO |
| C | ... KPRO |
| C | ... QPRO |
| FORMAT: FDT/p,where p=C,S,D,K,A,N,?. For more information n type "?FDT" | |

**Таблица 2**

| | | |
|---|---|---|
| File Definition | | Table for the base DOC |
| | | Last records number: 0 |
| | | Number of the data elements: 13 Key DE's 10 |
| K :Ii..Ki..: C | 4 | SC (shortcode) |
| K :Ie......: C | 5 | R (reference) |
| K :Ie......: C | 1 | TY <r> |
| K :Ie......: C | 1 | D <r> |
| :........: N | | AUTHORS (auth) |
| K :....Ke..: C | 4 | A <authors> |
| K :........: N | | SPOKESMAN (spok) <a> |
| K :........: C | 1 | I <authors> |
| :........: C | | T (title) |
| K :........: N | | REAC-DATA |
| K :IcOcKiQc: C | 40 | RE <reac-data> |
| K :........: R | 2 | PLAB <reac-data> |
| K :......Qc:-I | 1 | FSP |
| K :........: N | | PART-PROP |
| K :IcOcKiQc: C | 1 | P <part-prop> |
| K :Ie..Ki..: C | 1 | PP <part-prop> |
| K :........: C | 2 | DEN |
| K :........: C | 2 | DCH |
| FORMAT: FDT/p,where p=C,S,D,K,A,N,?. For more information type "?FDT" | | |

Рис. 2.3:

Рис. 2.4:

| **Таблица меток** | | |
|---|---|---|
| Команда | Функция | Примечание |
| % | любой символ | |
| | любая цепочка символов | ** "*" |
| $: n_1 - n_2 \ldots :$ | описание зоны контроля | |
| $< s_1 > s_2 \ldots$ | ограничивающее описание | >> ">" |
| | контролируемого символа | << "<" |
| # ..... # | описание запрета выбора (NOT) | ## "#" |
| $ | пересечение условий (AND) | $$ ≡ "$" |
| & | объединение условий (OR) | && ≡ "&" |

# Глава 3

# Язык описания данных

Язык описания данных определяется в BDMS на двух уровнях:

- ядром системы формируется основная часть описания структуры данных, что отражено внешне в документах FDT и правилах форматирования документов BDMS — баз данных;

- прикладными процессорами контроля данных, поддержива- ющими основной для PPDS язык: Particle Physics Data Language (PPDL), и, некоторые элементы структуры документа, не нашедшие отражения в ядре системы, например, описание таблиц.

В ядре системы только одна процедура ответственна за обработку вводимых данных - это подсистема "РЕДАКТОР". Подсистема функционирует как в диалоговом, так и в фоновом режиме. Возможен последовательный ввод данных документа и ввод данных с одновременным редактированием. Команды управляющие работой редактора являются составной частью языка описания данных. Первая часть этой главы посвящены описанию подсистемы РЕДАКТОР. Во второй приводится описание PPDL, как реализованной в PPDS возможности, предоставляемой для контроля данных прикладными процессорами.

## 3.1 Подсистема "РЕДАКТОР"

РЕДАКТОР это подсистема BDMS вызываемая EXECUTIV'ом для создания новой записи в памяти программы, прежде чем дописать ее к базе данных (инструкция ADD) или для модифицирования существующей записи, которая считывается в память из базы данных (инструкция MODIFY). Он вызывается также и при определении базы данных для создания или модифицирования FDT записи подсистемой DEFINE, и при загрузке документов в базу данных подсистемой LOAD.

Существуют две моды РЕДАКТОРа: мода дополнения, используемая при создании или расширении записи, и редактирующая, позволяющая выборочно заменять или модифицировать значения ЭД, вставлять или уничтожать генерации ЭД. Новая запись создается в моде дополнения, однако, можно перейти в редактирующую моду для исправления ошибок или устранения пропусков перед перемещением записи в базу данных. Таким образом, редактирование уже существующей записи обычно осуществляется в редактирующей моде а мода дополнения используется при расширении записи простым добавлением генераций ЭД.

РЕДАКТОР читает в свободном формате входную информацию, включающую команды и элементы данных. Команды сообщают ему, какие операции надо производить с данными, устанавливают операционную моду. Имена элементов данных при этом могут возникать как самостоятельно, так и в виде присвоенного значения 7) в зависимости от операции, определенной последней командой. В редактирующей моде имена элементов данных сопровождаются одним или более номером генерации, которые идентифицируют конкретный элемент данных в структуре записи. Выражение 7) присваивает значение генерации элемента данных.

### 3.1.1 Команды редактора

Редактор управляется командами которые либо устанавливают его операционную моду, либо немедленно производят какое-либо действие. Все команды можно использовать в односимвольном виде (в их списке при этом нет неоднозначностей), с пред- шествующей звездочкой и последующим пробелом.

#### Команды, устанавливающие оперативную моду

#### Мода дополнения

*Append добавление к записи очередной генерации элемента данных. Эта мода включена по умолчанию.

#### Редактирующая мода

*Replace замена значения элемента данных.

*Substitute обменять подстроку внутри элемента данных.

*Delete вычеркнуть генерацию элемента данных из записи.

*Insert ввести в запись генерацию элемента данных.

#### Команды, имеющие немедленное действие

** выход из редактора — запись будет размещена в базе данных.

*Cancel прервать редакцию - не помещать в базу данных новую или отредактированную запись.

*List распечатать на экране дисплея текущее состояние редактируемой записи.

*End конец редакции. Этой командой отмечается конец записи при вводе сразу нескольких записей.

### Мода дополнения. Порядок ЭД

В моде дополнения порядок ЭД в записи определяется порядком, в котором поступают ЭД и их связями с родителями. ЭД связанные с генерациями их родительских ЭД (или REC-ID для ЭД уровня записи) формируют упорядоченный список. В общем случае каждый узловой или значимый ЭД создает новую генерацию ЭД которая добавляется в конец списка генераций связанных с последним созданным родительским ЭД.

Например: если данные добавлялись к базе данных, описанной в 1.2,

```
AUTHORS.;
  A. = J.J.Aubert;
  A. = U.Becker;
  I. = MIT;
AUTHORS.;
  A. = Y.Y.Lee;
  I. = BNL;
```

будут созданы две группы авторов, первая содержит двух авторов из MIT, и вторая с одним автором из UCB.

Точка, следующая за именем ЭД означает, что это "следующий" ЭД в списке одноименных. Имя ЭД без точки означает, что это "первый" или ЭД с номером генерации 1. Следующие два подраздела этой главы поясняют смысл этого условия.

С другой стороны, генерации ЭД можно размещать в желаемом порядке, корректно придерживаясь только последовательности старший - подчиненный при этом порядок генераций ЭД в моде дополнения несущественен. ЭД для ссылки "R" может быть вставлен где угодно в предыдущем примере без какого-либо воздействия на описание группы авторов.

### Автоматическое создание родительского ЭД

В моде дополнения узловой родительский ЭД во входных дан- ных обычно опускается. Генерации таких ЭД необходимые для объединения подчиненных генераций ЭД будут созданы автомати- чески, если:

1. во входных данных встречается соподчиненный ЭД, а родительский еще не существует, или

2. встречается генерация 1 одного из подчиненных ЭД, т.е. имя ЭД использовано без следующей за ним точки а последний созданный родительский ЭД уже имеет как минимум одну такую же, связанную с ним генерацию ЭД.

Таким образом приведенный выше пример мог быть представ- лен в более простом виде:

```
A = J.J.Aubert;
A. = U.Becker;
```

```
I = MIT;
A = Y.Y.Lee;
I = BNL;
```

или, что эквивалентно:

```
I = MIT;
A = J.J.Aubert;
A. = U.Becker;
I = BNL;
A = Y.Y.Lee;
```

каждая генерация A в первом случае или I во втором приведет к созданию узлового родительского элемента данных AUTHORS.

Автоматическое создание родительского узлового ЭД будет происходить даже если родительский ЭД не описан как УЗЛОВОЙ. В этом случае будет создан ЭД с нулевым значением.

### Автоматическое создание записи

Если при добавлении записей ЭД уровня записи появляется более одного раза без "точки", то его вторая генерация включит процедуру размещающую предыдущую запись в базе данных и начнет НОВУЮ запись так же, как в случае редакторской команды *E. При этом вторая генерация ЭД уровня записи станет частью новой записи.

Таким образом саму запись можно рассматривать как чистый узловой родительский ЭД для всех ЭД уровня записи, автоматическое создание записи полностью аналогично автоматическому созданию узлового родительского ЭД описанного в предыдущем абзаце.

Когда редактор вызван инструкцией EXECUTIVE MODIFY автоматическое создание записи бессмысленно, поэтому в таком случае в моде дозаписи повторный ЭД без точки "следующий" рассматривается как ошибка во входных данных.

### Многократное присвоение значений ЭД

Несколько последовательно появляющихся генераций ЭД можно описать без повторения имени ЭД. Это делается указанием многократного присвоения значения ЭД одному имени, основная форма которого:

{имя ЭД} = {1 значение ЭД}; {2 значение ЭД}; ...

Предыдущий пример в этом случае выглядит более просто:

```
A = J.J.Aubert; U.Becker; I = MIT;
A = Y.Y.Lee; I = BNL;
```

или, в эквивалентной форме:

```
I = MIT; A = J.J.Aubert; U.Becker;
I = BNL; A = Y.Y.Lee;
```

Разница между двумя записями:

PLAB = 19.000,20.000;

и

PLAB = 19.000;20.000;

можно легко понять. В первом случае просто создается генерация ЭД PLAB чьим значением является двухкомпонентный вектор. Во втором создадутся две генерации ЭД PLAB имеющие соответственно значения 19.000 и 20.000.

**Выполняя многократное присвоение значения входной процессор будет пытаться интерпретировать все, что не распознается как имя ЭД или ЭД как следующее значение предыдущего ЭД. Если имя ЭД введено с ошибкой, ошибочное описаниеН ЭД не будет распознанно, и обработается входным процессором как следующее значение ранее введенного ЭД текстового типа. Как ошибка ситуация будет воспринята только если предыдущий ЭД не является текстовым.**

### Мода редактирования

Любая из инструкций редактора *Substitute, *Replace, *Delete или *Input переводит его в моду редактирования.

### Указание генерации жлемента данных

В редактирующей моде ЭД предназначенный к редактированию определяется по его положению в структуре. Чтобы точно указать генерацию необходимо задать имя элемента данных и номер генерации, номер родительской генерации с которой связана данная, номер генерации с которой связана родительская и т.д. вплоть до предка уровня записи. Серия номеров генерации приписывается к имени ЭД в таком порядке: т.е. в порядке возрастания удаленности родства, от ближних к предкам, как серия чисел отделенных от имени и друг от друга точкой.

Обратимся в качестве примера к структуре записи в части 1, A.4.1 будет идентифицирован как четвертый автор "A.4" первой группы авторов "AUTHORS.1".

В определенных условиях можно обойтись без указания всех этих номеров генерации так как редактор поддерживает систему описания по умолчанию, о чем будет рассказано ниже в разделе СТЕК ...

*Replace

Эта инструкция используется если необходимо присвоить новое значение существующей генерации ЭД. Например:

*R A.2.1 = T.Becker;

заменит значение ЭД A.2.1 на "T.Becker".

Бессмысленно использовать данную инструкцию для чисто узловых ЭД принципиально не имеющих присвоенного значения.

*Substitute

Эта инструкция используется для замены подцепочки символов в ЭД текстового типа. Инструкция имеет вид:

{s}{старая подцепочка}{s}{новая подцепочка}{s}

где "s" — символ разделитель, который может быть любым символом из тех, что не встречаются ни в старой ни в новой под- цепочке символов. Все пробелы используемые в подцепочках — значимы.

{новая подцепочка} заменит в значении ЭД первую встретившуюся в значении ЭД подцепочку символов совпадающую с отмеченной в {новой подцепочке}. Если {старая подцепочка} не задана, то первая встреченная в значении ЭД {старая подцепочка} будет просто удалена.

Например, если A.2.1 имеет значение "T.Baker", то инструкция

*S A.2.1=/ak/eck/;

заменит его на "T.Becker", инструкция

*S A.2.1=//U/;

заменит это значение на "UT.Becker", а инструкция

*S A.2.1=/T//;

заменит на "U.Becker;

Для контроля правильности редакции система печатает значение отредактированного ЭД.

*Delete

Эта инструкция используется для удаления ЭД указанного в параметре к ней, и всей ветви ЭД зависимых от удалаемого, например:

*Delete R.2;

будет удален ЭД R.2, TY.1.2 и D.1.2, связанные с ним.

После удаления генерации ЭД остальные генерации этого же ЭД связанные с тем же родительским ЭД и следующие за удаленным в списке ЭД, будут идентифицироваться на единицу меньшим номером.

*Insert

Эта инструкция используется вместе с описаниями ЭД или именами генераций ЭД для чисто узловых ЭД. Каждое из них будет вставлено в список ЭД в записи перед ЭД указанным в инструкции.

Например, если имя автора по ошибке было пропущено перед A.2 в AUTHORS.1 то его можно вставить инструкцией:

*I A.2.1 = U.Becker;

После вставки генерации ЭД у всех других сузествующих генераций этого ЭД связанных с тем же родительским ЭД и следующих за вставленным номер генерации увеличится на единицу. Иными словами, вставленная генерация будет иметь номер обозначенный в инструкции.

При добавлении генерации ЭД после всех уже существующих генераций этого ЭД связанных с одним и тем же родительским ЭД можно специфицировать номер генерации больший, чем уже существующие или указывать номер 0, или просто не указывать номера, но не забывать проставить точку. Таким образом, приведенная выше инструкция будет иметь тот же эффект даже если ЭД AUTHORS.1 имеет только один подчиненный ЭД.

Множественное присвоение значений, следующее за командой *Insert введет несколько новых генераций перед специфицированной в инструкции. Они создадутся в записи в порядке их следования в операторе присвоения. После ввода генерации ЭД связанные с одним родителем и следующие за введенными генерациями этого ЭД будут иметь номер генерации увеличенный на число вставленных ЭД.

Например:

*I A.3.1 = P.J.Biggs; J.Burger;

дополнит список AUTHORS.1 A.3=P.J.Biggs; и A.4=J.Burger. Если A.4 уже существовал до ввода новых ЭД то его номер генерации станет A.6, и т.д.

### Порядок редактирования

Один момент предыдущих обсуждений будет полезен в дальнейшем рассмотрении: номера генераций ЭД относительны, зависят от положения в списке и могут меняться в результате редактирования. После того, как генерации ЭД вводятся или удаляются, генерации следующие за ними идентифицируются номерами большими или меньшими, чем до редактирования.

Чтобы избежать ошибок, особенно при неинтерактивной работе, первыми надо делать операции замены значений в уже существующих ЭД, а введение новых ЭД и удаление ЭД проводить "снизу-вверх", т.е. первыми должны редактироваться генерации ЭД с наибольшими номерами. В таком случае редактируемые ЭД сохраняют свои номера.

При интерактивной работе текущие значения номеров генерации можно уточнять с помощью команды *List.

### Стек памяти

В редактирующей моде редактор позволяет использовать систему задания номеров генерации в иерархии записи "по умолчанию", допускает не полное описание пользователем позиции ЭД в записи.

Редактор запоминает "путь" пройденный им в структуре записи до редактируемого элемента. Затем, если положение следующего редактируемого элемента задано не полностью, он пытается определить положение по запомненной части пути. Если это допустимо, то координата в структуре определится ранее заданной и вновь задаваемой частями ее описания.

Запись "пути" в структуре забывается при переходе в моду дозаписи.

Лучше всего понять действующий механизм на примерах.

1. Пусть требуется заменить значения нескольких A в блоке AUTHORS.2 Этого можно добиться с помощью инструкции:

```
*R A.2.1= U.Becker;
   A.3  = P.J.Biggs;
   A.4  = J.Burger;
```

В примере не специфицируется номер генерации родительского ЭД AUTHORS.1 для второго и третьего ЭД.

2. Предположим, что была пропущена целиком группа авторов. Ее можно вставить инструкцией:

```
*I AUTHOR.1;
    A = U.Becker; P.J.Biggs; J.Burger; M.Chen;
    I=MIT;
```

До тех пор, пока номер родительского элемента не специфицируется, он будет выбираться из заданного ранее.

## 3.2 Особенность языка PPDS данных

### 3.2.1 Тезаурус PPDL

PPDL задается терминологическим словарем (тезаурусом), правилами синтаксиса и, кроме того, связан со структурой информации в базах данных. Язык описания данных используется для проверки информации на вводе и служит средством для формирования запроса пользователя. В зависимости от требований к базе данных язык может быть в большей или меньшей степени формализован. Кроме того, язык описания данных представляет собой самостоятельную ценность, т.к. содержит описание предметной области в сжатом и агрегированном виде.

PPDL содержит словари и правила синтаксиса для всех основных терминов и, таким образом, позволяет осуществлять жесткий контроль информации при вводе и точный отбор при поиске.

Термины PPDL максимально приближены к естественному языку, принятому в публикациях по физике высоких энергий (большинство терминов являются лексическими свертками обозначений и описаний). Соответствие вводимой информации тезаурусу PPDL и его синтаксические правила проверяются пакетами прикладных программ при вводе информации в базы данных.

Термины и синтаксис PPDL разделены на несколько частей, соответствующих описанию разных типов данных (реакция, библиографическая ссылка, ускоритель, детектор, описание наблюдаемых величин и т.д.). В разных базах данных используются различные части PPDL, но при этом общие для разных баз данных описываются одинаково (например, реакция, ссылка). В PPDL можно вводить синонимы к терминам. Если у термина существуют синонимы, при вводе в базу данных его значение заменяется одним из них (наиболее старшим, расположенном в списке раньше других). Этот механизм используется для изменения терминов. Кроме того, это обеспечивает пользователям дополнительное удобство при поиске: свой запрос они могут формировать в старых терминах, не заботясь о возможных изменениях тезауруса. Текущая версия тезауруса хранится в базе данных VOCABULARY с встроенным описанием и может использоваться как интерактивно, так и в виде полной распечатки упорядоченных списков терминов с их описанием.

#### Описание синтаксиса и правил формирования словарей PPDL

**<u>Институты</u>**

Как правило, термины, описывающие институты, — это четырехбуквенные сокращения их названий или акронимы городов в которых институты расположены. В распечатке тезауруса присутствует алфавитно-упорядоченный по городам и по странам список терминов. Такое представление облегчает поиск нужного термина при формировании запроса и дополнении тезауруса.

**Ускорители**

Термин, описывающий ускоритель, состоит, как правило, из описания института и возможного дополнения, если ускорителей в институте несколько (например CERN, CERN-SPS, CERN-ISR...) Для циклотронов используется дополнение - CYC (например LBC-CYC).

Более не используемые линейные ускорители и синхротроны описываются так же, как и действующие установки.

Эксперименты на космических лучах описываются как проведенные на "ускорителе" COSM. Если источник излучения локализован, то к сокращению COSM добавляются поясняющие символы, например COSM-SN1987A; COSM-SUN; и т.д. В PPDS включены только те эксперименты на космических лучах, которые дают информацию о свойствах частиц и их взаимодействиях.

Редкие случаи (в основном связанные с поиском новых частиц) при которых не используются ни ускорители, ни космические лучи, отмечаются как проведенные на "ускорителе" NONE.

**Детекторы**

Термины формируются на основе названий детекторов, введеных создателями. Если детектор не имеет собственного названия, используются описания типов методики, применяемой в эксперименте. Для пузырьковых камер используется следующая структура термина:

<тип>-<описание.1>-<описание.2>

<тип> может быть:

HBC – водородная ПК;
DBC – дейтериевая ПК;
HEBC – гелиевая ПК;
HLBC – ПК с тяжелым наполнением.

<описание.1> — размеры камеры или ее собственное имя;

<описание.2> — особенности конструкции:

HYB – гибридная система;
RAP – быстроциклирующая ПК;
TST – трекочувствительная мишень.

В распечатке тезауруса представлено два списка детекторов: упорядоченный алфавитно и по институтам, в которых детекторы установлены.

**Типы ссылок**

Термины формируются на основе лексических сверток, например:

JOUR – журнальная ссылка;
PREP – препринт и т.д.

**Журналы**

Журнальные ссылки в языке PPDL описываются в фиксированном формате:

<код журнала>␣<том>,␣ <страница>

Коды журналов являются аббревиатурами их названий.

**Цитируемая литература**

В некоторых случаях записи баз данных PPDS содержат описание литературы, цитированной в публикации.

Для этого, в общем случае, используется следующий формат

<тип цитирования> *<описание ссылки>

Тип цитирования описывает связь данной публикации с цитированной. Например:

| | | |
|---|---|---|
| MORE | – | данная работа служит продолжением цитированной. |
| S | – | результаты данной работы полностью заменяют результаты цитированной и т.д. |

**Частицы**

Термины, описывающие частицы, соответствуют описаниям, принятым PDG(US) для RPP (Review of Particle Properties). Во всех терминах, кроме тех, которые используются для хорошо известных частиц, в скобках указана номинальная масса (например, F1(1410)). Для многих частиц в термине указан заряд: PI+, PI-, PI0. Для барионных резонансов используются спектроскопические обозначения N(1440P11)+. Если квантовые числа резонанса не установлены, вместо них указывается символ "B": SIGMA(1690B)+. Античастицы содержат после термина дополнительный набор символов "BAR". Заметим, что в терминах для античастиц указывается реальный знак:

DELTABAR(1232P33)–.

Кроме терминов, соответствующих частицам и резонансам, в словаре частиц присутствуют термины, описывающие гипотетические состояния, методические особенности, классы конечных состояний. Для удобства пользования и сопровождения весь тезаурус разбит на части, соответствующие следующим классам терминов:

1. частицы и резонансы из RPP (надежно установленные);

2. частицы и резонансы из RPP, нуждающиеся в подтверждении;

3. не подтвердившиеся состояния;

4. имена изотопических мультиплетов;

5. имена C-мультиплетов (объединяют частицы с античастицами);

6. гипотетические состояния;

7. описание методики (например: X-инклюзивный опыт);

8. ядра — из-за исторически сложившегося в литературе пересечения имен частиц и ядер некоторые имена ядер в PPDL не совпадают со стандартными.Предпочтение отдано частицам из-за большей частоты использования;

9. некоторые специальные группы состояний (например: BARYONIUM, TOPONIUM и т.д.) Чаще всего это термины объединяющие состояния, соответствующие различным возбуждениям;

10. классы состояний по ароматам (CHARM, STRANGE и т.д.) и типам взаимодействий (LEPTON, HADRON). Адроны, в свою очередь, классифицируются по кварковому составу (MESON, BARYON, EXOTIC). Сюда же относятся более сложные конструкции: CHARMED-BARYON и т.п.;

11. термины, используемые для описания неопределенного числа частиц в конечном состоянии:

| | | |
|---|---|---|
| partS | – | 2 или более частиц; |
| part(S) | – | 1 или более частиц; |
| (partS) | – | 0 или более частиц; |
| (part) | – | 0 или 1 частица; |
| .LT.npart | – | менее n частиц; |

    Здесь part — термин из предыдущих классов. В этом же классе содержатся термины типа MULT(part), используемые для описания распределений по множественности.

В распечатке тезауруса частиц присутствует несколько списков терминов: упорядоченный по алфавиту общий список, упорядоченные по алфавиту списки отдельных классов, а также список терминов, соответствующих состояниям указанным в RPP, упорядоченный по массе и квантовым числам (порядок аналогичен применяемому в RPP).

Термины из словаря частиц являются основой для описания реакций. Реакция на PPDL записывается в следующем виде:

<начальное состояние>_ –>_<конечное состояние>

Описатель начального состояния формируется или из имени распадающейся частицы (при описании распада) или из имен частиц пучка и мишени, разделенных пробелами, при описании реакции столкновения.

Описатель конечного состояния, в простейшем случае, состоит из имен частиц в конечном состоянии, разделенных пробелами. Например:

PI- P –> PI0 N

Если в конечном состоянии присутствует несколько тождественных частиц, перед соответствующим именем указывается их количество:

PI- P –> N 2PI+ 2PI-

Если изучалось объединение нескольких конечных состояний, это записывается в виде

PI- P –> SIGMA0 K0 + LAMBDA K0

Эта же реакция может быть записана короче, с использованием скобок:

PI - P –> ( SIGMA0 + LAMBDA ) K0

Аналогично знак "-" используется для указания исключенных из рассмотрения состояний. Для сокращения записи можно использовать конструкцию типа:

PBAR P –> PBAR N PI+ + CC,

где "CC" соответствует зарядово-сопряженному состоянию. Если необходимо описать распад частицы, описание продуктов распада помещается за описанием частицы и заключается в скобки "<>".

PI+ P –> SIGMA(1385P13)+ < LAMBDA < P PI - > PI+ > K+

Заметим, что при вводе информации в базы описатели реакций преобразуются в фиксированный внутренний формат, причем частицы в конечном состоянии упорядочиваются и приводятся подобные. Аналогичным образом обрабатывается запрос пользователя. Поэтому при поиске имена в описателе конечного состояния можно писать в произвольном порядке. При этом наличие пробелов между именами и разделителями

–> ( ) < > + -

является обязательным.

При работе с описателями реакций часто возникает необходимость поиска по начальному или части конечного состояния. В некоторых базах данных при вводе информации прикладные программы анализируют описание реакции и генерируют в записи соответствующие элементы данных. Независимо от этого существует возможность поиска такого типа без использования дополнительных элементов данных. В последнем случае применяется последовательный просмотр списка ключевых выражений, соответствующих описателям реакций, и анализ каждого из них на соответствие запросу. Запрос оформляется следующим образом:

DE > <описатель части конечного состояния> или

DE < <описатель начального состояния>,

где DE — имя элемента данных, содержащего описание реакции. Символы "<" и ">" в этом случае имеют не буквальный смысл, а служат указанием на нестандартный способ сравнения. Например:

RE > MU+ E-;

RE < PBAR P;

При поиске по начальному состоянию возможно применение комбинации "UNSPEC" в качестве символа "любой".

Классы реакций: термины, описывающие реакции по конечному состоянию.

### ЭКСКЛЮЗИВНЫЕ реакции

Если в конечном состоянии в реакции представлены только стабильные частицы, и при этом все они идентифицированы, то такая реакция относится к классу эксклюзивных. (EXCL).

При тех же условиях если в конечном состоянии в реакции исследуется резонансное состояние, то реакция принадлежит классу EXRP (Exclusive Resonance Production).

(Exclusive Partial Sum) EXPS — эксклюзивные реакции, в которых экспериментальная методика не позволила разделять конечные состояния. В этом случае неразделяемые конечные состояния описываются в реакции совместно, как "сумма" конечных состояний. Различные конечные состояния отделяются знаком "_ + _".

### ИНКЛЮЗИВНЫЕ реакции

Реакции в которых идентифицирована только часть конечного состояния а все остальные нерегистрируемые частицы отмечены специальным кодом "X" или "ANYTHING". Инклюзивные реакции: INCL.

Классы инклюзивных реакций аналогичны классам эксклюзивных: INRP, INPS.

### ПОЛУИНКЛЮЗИВНЫЕ реакции.

В случае когда недетектируемая часть конечного состояния состоит из нейтральных частиц, ее отмечают одним из имен: NEUTRAL(S), NEUTRALS, (NEUTRALS). Смысл этой кодировки в полуинклюзивных реакциях такой же, как у кода ANYTHING в инклюзивных. Для класса полуинклюзивных реакций используется имя SEMI. Классы SERP и SEPS введены аналогично EXRP и EXPS.

### ДВАЖДЫ ПОЛУИНКЛЮЗИВНЫЕ реакции

Если просуммированы кроме нейтральных фиксированное количество заряженных треков, то используется код "nCHARGED (NEUTRALS)", где n — число заряженных треков. Этот код используется так же, как "ANYTHING".

Имена классов дважды полуинклюзивных реакций: DSEM, DSRP, DSPS.

Все остальные реакции объединяются общим классом: MISC (вне классификации).

**Дескрипторы данных**

В базах данных PPDS используются две формы описания данных: краткая (для библиографических баз данных) и детальная (для фактографических баз). Соответственно этим формам существуют два словаря и два набора правил синтаксиса.

**Краткие дескрипторы данных**

Краткие дескрипторы данных соответствуют широким классам данных и поэтому дают лишь общее представление о числовой информации, представленной в публикации. Например:

ANGP — угловое распределение продуктов реакции,

MASS — спектры масс и т.д.

Для описания двумерных распределений используются более сложные конструкции:

MASS*MASS, ANGP*PT и т.д.

**Детальные дескрипторы данных**

Детальные дескрипторы данных содержат полную информацию о полученных сечениях, распределениях и т.д. Общая структура этих дескрипторов выглядит следующим образом:

<основное имя>(<имя аргумента>=<значение аргумента>, ...)

Основное имя служит для описания типа наблюдаемой величины. Например:

| | | |
|---|---|---|
| SIG | – | сечение, |
| THETA | – | угол, |
| PT | – | поперечный импульс и т.п. |

Для более детального описания данных основные имена снабжаются дополнительными описателями-аргументами. Наиболее употребительными аргументами являются:

**P** – указание частицы, для которой приводится наблюдаемая величина. Значением аргумента в этом случае является порядковый номер частицы или набора одинаковых частиц в записи реакции;

**RF** – описание системы отсчета, возможные значения аргумента: CM, LAB, N; где "N" — номер частицы в реакции. RF=3; например означает, что процесс рассматривается в системе покоя частицы 3.

**XYZ** – описание системы координат. Возможные значения аргумента: SH(s-канальная спиральная система, TH (t-канальная спиральная система), ADAIR (система Адера).

Так например запись

XL(P=3, RF=LAB, DEF=PL/PMAX)

означает, что рассматривается распределение первой частицы из записи конечного состояния реакции по продольной масштабной переменной в лабораторной системе. Определение масштабной переменной представлено аргументом DEF.

Детальные дескрипторы данных могут образовать сложные конструкции с использованием операторов:

+ , - , * , ? , **

и функций:

SIN, COS, ABS и т.д.

Кроме того, используется синтаксис для описания дифференцирования и интегрирования:

D(SIG)/D(PT)

D2(SIG)/D(XL)/D(PT)

INT(E*D3(SIG)/D3(P))*D(PT)

Тезаурус детальных дескрипторов данных содержит алфавитно-упорядоченные списки основных имен с пояснением их смысла и указанием возможных аргументов, а также списки имен аргументов.

**Дескрипторы свойств частиц**

Этот тип дескрипторов аналогичен кратким дескрипторам данных: он применяется в библиографических базах данных и служит для описания изучавшихся свойств частиц.

| | | |
|---|---|---|
| EX | – | поиски состояния, |
| PW | – | парциальные ширины, |
| MASS,M,DM | – | масса, |
| WIDTH,W | – | полная ширина, среднее время жизни, |
| MOM,MM | – | электрический (магнитный момент), отношения моментов; |
| DEC | – | параметр распада при слабых или электромагнитных |
| QN | – | взаимодействиях квантовые числа; |

**Единицы и масштабы**

Значение элемента данных описывающего имя измеренной в работе величины обычно имеет в PPDL следующую форму:

<имя величины>_IN_<единица измерения>

В таблице приведен список основных имен. Часть из указанных имен могут использоваться с предшествующим символьным описателем показателя степени 10, как также указано в таблице.

Таким образом, миллибарн — MB, миллион электронвольт — MEV, сантиметр — CM, и т.д.

Накоторые из этих имен могут быть представленны в виде арифметических выражений в формате фортрановских: MEV**2 = MeV**2, MB**.5 is "sqrt(mb)".

В описании единиц сохранена фортрановская конвенция на иерархию операторов и порядок операций.

ОПИСАНИЕ БАЗЫ ДАННЫХ VOCABULARY

Как уже указывалось, все словари PPDL с соответствующими описаниями смысла терминов хранятся в базе данных под управлением BDMS, что представляется удобным для пользования словарями и их сопровождения . Обращение к базе осуществляется через макрокоманду PPDS/3/.

В базе VOCABULARY хранятся различные словари и использование элементов данных, вообще говоря, зависит от типа словаря. Основные элементы данных, общие для всех словарей, описаны ниже:

USAGE — служит для разделения базы на отдельные словари.

Возможные значения:

| | | |
|---|---|---|
| I | – | институты; |
| AC | – | ускорители; |
| R | – | журналы; |
| TY | – | типы ссылок; |
| RR | – | типы цитирования; |
| P | – | частицы; |
| M | – | классы реакций; |
| DD | – | краткие дескрипторы данных; |
| PP | – | дескрипторы свойств частиц; |
| O | – | детальные дескрипторы данных; |
| A | – | аргументы детальных дескрипторов данных. |
| DE | – | детекторы; |
| U | – | единицы измерений; |
| S | – | шкала единиц измерений. |

| | | |
|---|---|---|
| ABBREVIATION | – | содержит термин. Появления этого элемента данных с номером больше единицы соответствуют синонимам термина. |
| COMMENT | – | описание термина. |

База VOCABULARY снабжена набором обслуживающих и прикладных программ:

- процессоры проверяющие новые термины при занесении их в базу данных VOCABULARY. Загрузочный модуль BDMS содержащий эти процессоры: BDMS$MAIN:VOC.EXE, тексты к нему размещены в директории VOC структуры PROC.

- программа, формирующая файл, который содержит полный список терминов языка и используется прикладными программами всех баз данных PPDS для проверки информации при вводе и при формировании запроса. Эта программа вызывается при работе с базой данных процессором BDMS$MAIN:VOC.EXE в EXECUTIVE контексте при использовании инструкции BDMS:

USER

результат ее работы записывается в файле PPDL.VOC.

- программа распечатывающая тезаурус PPDL и описание синтаксиса языка, размещенная в директории VOCBOOK структуры TOOLS дистрибутива.

PPDL как язык описания данных формировался параллельно с созданием библиографических и фактографических баз данных PPDS и претерпел за время своего существования значительные изменения. Одни из этих изменений явились следствием развития физики частиц, другие служили целям совершенствования языка. Несмотря на то, что в настоящее время PPDL относительно стабилизировался, в нем продолжают происходить изменения. В языке широко применяется механизм синонимов и поэтому изменения, как правило, не приводят к дополнительным затратам на редактирование баз данных и не вынуждают пользователя следить за всеми редакциями словарей. В центрах, сотрудничающих в сопровождении баз данных, могут использоваться слегка различающиеся версии языка, так как строгое согласование словарей ведет к задержке в обработке публикаций.

Вследствие частых модификаций языка публиковать тезаурус представляется нецелесообразным. Распечатка последней версии тезауруса производится по требованию пользователя.

| **Список основных имен.** | | |
|---|---|---|
| EV | - | электронвольты |
| MPI | - | массы пиона |
| MCHARPI | - | в массах заряженного пиона |
| MNEUTPI | - | в массах нейтрального пиона |
| MPROTON | - | в массах протона |
| RAD | - | радианы |
| DEG | - | градусы |
| SR | - | стерадианы |
| L | - | литры |
| SEC | - | секунды |
| M | - | метры |
| EVENTS | - | события |
| B | - | барны |
| PCT | - | проценты |
| PI | - | p = 3.141562.... |
| ? | - | произвольная единица |

| **Список показателей степеней 10.** | | |
|---|---|---|
| E | 10**18 | eksa- |
| PE | 10**15 | peta- |
| T | 10**12 | tera- |
| G | 10**9 | giga- |
| M | 10**6 | mega-(как приложение к eV) |
| MEGA | 10**6 | mega-(во всех остальных случаях) |
| K | 10**3 | kilo- |
| H | 10**2 | hekto- |
| DA | 10**1 | deca- |
| C | 10**(-2) | centi |
| MILLI | 10**(-3) | milli-(как приложение к eV) |
| M | 10**(-3) | milli-(во всех остальных случаях) |
| MU | 10**(-6) | micro- |
| N | 10**(-9) | nano- |
| P | 10**(-12) | pico- |
| F | 10**(-15) | femto- |
| A | 10**(-18) | atto- |

# Глава 4

# Язык управления системой

Управление системой, управление поиском, преобразованием и выдачей данных осуществляется с помощью инструкций системы. Инструкции BDMS разбиты на несколько категорий-контекстов, реализуемых соответствующими подсистемами BDMS. В BDMS ИНСТРУКЦИЯ — это одна из допустимых в текущем контексте КОМАНД доопределенная квалификаторами и параметрами. Набор допустимых команд определяется подсистемой. Формат инструкции фиксирован в пределах подсистемы. Инструкции одна от другой отделяются пробелом. Количество пробелов не существенно. Если требуется использовать пробел в пределах одной инструкции, например в тексте описания данных в параметре, то такой текст заключается в кавычки.

Инструкции BDMS при вводе могут следовать одна за другой, без перехода на следующую строку. При переходе на следующую строку инструкция может оказаться разорванной, если переход произведен раньше, чем закончено описание инструкции, или "закончилась" входная строка, размер которой по умолчанию равен 80. В этом случае к введенным символам в строке добавляется символ пробел.

Основной формат ИНСТРУКЦИИ BDMS: КОМАНДА — английский глагол (например FIND, LIST, DELETE), квалификаторы команды, отделяемые один от другого и от команды косой чертой "/", список параметров отделенных друг от друга и от команды или ее квалификаторов запятыми:

КОМАНДА/квалификатор1/...,параметр1,...

В разделе 3 рассматривалась подсистема редактор с отличным от 4.1 форматом инструкции:

*КОМАНДА␣[данные;]

Здесь, в нарушении основного формата внутри инструкции использован пробел. Другой пример: формат описания запроса (см главу 2.).

Основной набор команд подсистемы можно можно изменять сокращая список команд или расширяя его вводя новые (как описано в следующем разделе), но при этом сохраняется формат команды.

## 4.1 Интерпретатор команд

При работе с BDMS пользователь в каждый момент времени взаимодействует с одной из ее подсистем, например: DEFINE, LOAD, EXECUTIVE, GUIDE, MODIFY. В общем случае подсистемы различаются списком команд.

В версии 4.0 процедура распознавания команд и некоторые общие процессы обработки команды пользователя сведены в одну общую процедуру: ИНТЕРПРЕТАТОР КОМАНД. Это позволило ввести простые правила создания самоописаний к системе, ввести процедуры, упрощающие манипулирование командами: введение синонимов к командам, макрокоманд.

При ошибках пользователя в написании команд, или по его распоряжению система печатает на экран дисплея список команд подсистемы. Команды в этом списке алфавитно упорядоченны.

Интерпретатор команд допускает сокращение имени команды до возникновения неоднозначности при сравнении с сокращенными именами других команд. В случае такой неоднозначности в интерактивном режиме система предупреждает пользователя об ошибке, печатая список неоднозначно определяемых команд при таком сокращении.

Интерпретатор считает командой группу символов в которую не могут быть включены:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ( ) | – | пробел, | (/) | – | косая черта, | (,) | – | запятая, |
| (.) | – | точка, | (:) | – | двоеточие, | (;) | – | точка с запятой. |

а так же, символы зарезервированные в качестве подкоманд ИНТЕРПРЕТАТОРА:

```
(@)          (=)              (#)
(&)          (:)              (?)
```

Пробел в общем случае отделяет друг от друга инструкции. Параметры команды отделяются друг от друга запятой, квалификаторы команды выделяются символом "/", этим же символом они отделяются друг от друга.

## 4.2 Подкоманды инерпретатора команд

В качестве подкоманд приняты символы, не использующиеся в качестве первого символа команды:

```
"@",  "=",  "#", "?", "&", "+"
```

@{имя файла} — чтение входных данных из файла {имя файла};
@/ — переход на уровень системного монитора;
? — обращение к процедурам самоописания системы;
= — введение синонима к существующей команде;
# — введение описания макрокоманды;
& — подавление существующей команды из списка команд;
+ — восстановление вычеркнутой команды в списке.

Параметрами подкоманд интерпретатора команд являются команды соответствующей подсистемы.

Синтаксис языка интерпретатора команд: подкоманды односимвольные, первый параметр вводится сразу за символом подкоманды, без разделителей.

? — вызов подсказки

При интерактивной работе, во всех случаях ожидания системой реакции пользователя, возможны консультации:

| | | |
|---|---|---|
| ? | – | распечатывается список всех команд подсистемы: основных (базовых) команд, синонимов к ним и макрокоманд; |
| ?# | – | распечатывается только список синонимов к базовым командам и макрокоманд подсистемы; |
| ?? | – | распечатывается список команд подсистемы с кратким поясняющим описанием каждой команды. |
| ?{команда} | – | подробное описание указанной команды. |

= — синонимы к основным командам

Инструкция

={команда1}={команда}

означает, что в дальнейшем в текущем сеансе можно использовать команду команда1 в качестве синонима команде {команда}. Интерпретатор команд не позволит определить в качестве синонима имя уже существующей команды. В этом случае будет передано сообщение:

Not unique new user-dependent name: "команда1"

Type new name:

модификация набора команд

| | | |
|---|---|---|
| &<команда> | – | закрывает доступ к дальнейшему использованию указанной команды; |
| +<команда> | – | открывает доступ к использованию ранее закрытой команды. |

# — определение макрокоманды

В BDMS/4.0 возможно объединить несколько команд в одной команде (макрокоманде), использующей параметры. Макрокоманда создается следующей конструкцией:

#<МАКРОимя>,"<определение макрокоманды>"

где <МАКРОимя> — уникальное для списка команд данного уровня имя новой команды. Место для "включения" параметров в <определении макрокоманды> отмечается знаком "макрокоманде параметры отделяются друг от друга и от имени макрокоманды запятой.

Пример определения:

```
#SEARCH,"FIND SB=%;** IF,NREC.LE.10:GOTO:LABEL LIST,%. LABEL"
```

Пример использования:

SEARCH,VVE,SC

Эта команда будет исполняться как следующая последовательность команд:

FIND SB=VVE;**

IF,NREC.LE.10:GOTO:LABEL

LIST,SC.

LABEL

Используя подкоманды интерпретатора можно, в принципе, полностью заменить базовый набор команд на другой, изменяя через введение синонимов имена команд, вводя составные команды. Однако, заниматься такой деятельностью при интерактивном режиме в каждом сеансе работы с базой данных накладно. В BDMS/4 введен ИНИЦИИРУЮЩИЙ ФАЙЛ: {имя базы данных}.LGF, который может содержать любые инструкции BDMS/4. Этот файл используется системой как входной командный файл при старте каждого сеанса связи с соответствующей базой данных. Система ищет файл с указанным именем сначала в директории пользователя, а затем (если не найдет), в системной: BDMS$COMPAS.

## 4.3 Вспомогательный набор команд

При диалоге с системой некоторые инструкции могут понадобиться в любом контексте. Таковы инструкции вызова вспомогательных описаний, FDT, INDEX, EDITOR, инструкции манипулирования выходными файлами, и т.д.

Команды к этим инструкциям объединены в доступную в любом контексте группу команд. Система переводится в контекст общего уровня если ПЕРВЫЙ символ в текущей строке @. Если вторым символом не является символ "/", то следующая за "@" цепочка символов воспринимается как имя файла содержащего инструкции системе. Канал ввода будет переключен на этот файл до тех пор, пока не завершится чтение информации из него или в нем не встретится инструкция STOP, терминирующая работу подсистемы. По завершении чтения файла система вновь переключает канал чтения входных данных, на тот, к которому была подключена. (Допускается последовательное переключение управления от файла к файлу с глубиной вложения равной трем)

Список вспомогательных команд можно распечатать на экране дисплея в интерактивном режиме инструкцией:

??

как и для списка команд любой подсистемы BDMS.

Часть команд вспомогательного набора представлены в подсистеме EXECUTIVE, где они обсуждаются подробнее.

### 4.3.1 Манипулирование файлами

Команды манипулирования файлами позволяют решать следующие задачи:

- просмотреть содержимое текстового файла, как внешнего, так и присоединенного к системе;
- присоединить к стандартному каналу вывода BDMS любой файл в моде дозаписи;
- скопировать в файл содержимое другого файла или дописать вручную;
- переименовать файл присоединенный к системе;
- отсоединить файл от стандартного канала BDMS;

Файлы распознаются по именам: внешние имеют стандартные имена в системе VMS. Для файлов BDMS МОНИТОР использует логические имена каналов: AUDIT, LOG, PRINT, DUMP, DOCUMENT. Различение этих двух ситуаций производится по небуквенному символу "." в параметре, обязательному при указании внешнего файла. По умолчанию все команды обращаются к каналу DOCUMENT.

@/LD,{имя файла}

Эта инструкция используется для распечатывания на экране дисплея списка файлов из указанной в параметре директории.

Примеры:

@/LD,[USER]FILE.DAT;3

или

@/LD,*.DAT;*

Список присоединенных к системе BDMS файлов можно визуализовать с помощью инструкции:

@/SHOWSYSTEM

Кроме списка файлов на экране будет отображено состояние задания, имена процессора и базы данных, имя пользователя и некоторые другие параметры. При работе в режиме MAIN этот список при первой записи в основные файлы базы будет переписан в специальный защитный файл, используемый для контроля фатальных сбоев системы.

@/APPEND[,chann][,FILEname]

присоединить к выходному каналу системы chann файл <FILEname> для дозаписи в него. Используются следующие имена выходных каналов: AUDIT, DOCUMENT, PRINT, DUMP, LOG. По умолчанию файл будет присоединен к каналу DOCUMENT. Если не указано имя файла, то будет приписан файл, определяемый для указанного канала по умолчанию. Например, для канала LOG файл с именем: {имя базы данных}.LOG.

@/INPUT[,][channel][,параметры вывода]

ввод данных с терминала в файл который определяется по наличию в параметре channel символа ".", или канал BDMS, по умолчанию DOC.

В списке параметров вывода используются те, что осмысленны в данном контексте, например — TOC/терминирующий текст/, или LI,{n} — число вводимых строк. По умолчанию (если не использован параметр или не указано число вводимых строк), ввод прерывается посылкой контрольного символа "^ Z"

@/RELEASE,<CHANN>,<FILEname>

Освобождает указанный выходной канал и присоединяет к нему файл с именем FILEname, или старым именем, если второй параметр опущен или неопознан как имя файла.

@/RENAME,OLDname,NEWname

Переименовывает существующий файл внешний, или приписанный к выходному каналу BDMS. Во втором случае OLDname можно указать именем канала.

@/LF[,имя][,параметры вывода]

печать на экран дисплея файла данных, содержащихся в файле приписанном к каналу "имя" (по умолчанию DOC), или, если в описании "имя" есть точка, то печать из файла с таким именем.

@/AUDIT (@/NOAUDIT)

дублировать (не дублировать) входную информацию (инструкции, данные) в файле <DBname.AUD>

@/LOG (@/NOLOG)

дублирует (не дублирует) инструкции пользователя в файл <DBname.LOG>.

### 4.3.2 Почта

@/MAIL

```
@/MAIL,<BOXNAME>,<DESTINATION>,<PROCESS>

       |PPDL    | |VOC|
       |SYSTEM  | |EXP|
       |......  | |DOC|  |W|    = WRITE
@/MAIL,|MANAGER|,|...|,*|C|    = CATALOGUE
       |USER    | |RD |  |R|    = READ
```

Пример:

| | | |
|---|---|---|
| @/MAIL,MANAGER,DOC,*W | – | пользователь желает передать сообщение администратору базы данных DOCUMENTS; |
| @/MAIL | – | пользователь желает ввести параметры к команде MAIL используя "меню". Система последовательно опросит пользователя:<br>=== Select box: ><br>после выбора "почтового ящика" аналогично будет выбран адресат и процедура. |

Система оперирует только с посланиями адресатам описанным в файле

[...MAIN.MAIL]STRUCTURE.DAT.

Процедуры:

| | | |
|---|---|---|
| CATALOGUE | – | каталог писем. Печатается только первая строка сообщения автоматически создаваемая системой: дата время имя_базы пользователь [его системные координаты] |
| READ | – | ознакомление с сообщением и вычеркивание его. |
| WRITE | – | запись послания. |

При записи система приглашает пользователя печатать следующую строку сообщения:

>

Реакцией пользователя может быть либо текст сообщения, либо команда системе. Команды могут быть следующие:

| | | |
|---|---|---|
| ? | – | печать информации о командах системы на данном уровне; |
| Copy,from | – | дополнить сообщение пользователя информацией из файла "from" который может быть как внешним, неприсоединенном к системе, так и системным (audit, log, doc, ...); |
| Edit,<EDITOR> | – | пользователь желает воспользоваться для подготовки послания одним их системных редакторов. По умолчанию — CONFED. |
| List | – | распечатать на экране уже записанную часть сообщения. |
| * | – | конец послания. |

В процедуре WRITE используется контроль орфографии двух языков, русского и английского.

### 4.3.3 Параметры системы

@/LP

Инструкции пользователя системе могут содержать числовые параметры:

| | |
|---|---|
| длина строки | LINE |
| число значащих цифр | SIGNIFICANCE |
| точность представления в выходном формате | PRECISION |
| число рекордов в выборке | NREC |
| число выборок | NSET |
| мода выдачи в формате DUMP | MODE |

Если параметр указан в инструкции пользователя с ошибкой, система дает пользователю сообщение об ошибке и подставляет вместо указанного значение по умолчанию.

По инструкции:

@/LP

система печатает на экран дисплея информацию о текущем статусе перечисленных параметров:

List of parameters:

| | min | default | max | mark | current |
|---|---|---|---|---|---|
| Line length | 10 | 80 | 132 | 0 | 80 |
| Figures | 1 | 6 | 6 | 0 | 6 |
| Dump modes | 1 | 3 | 4 | 0 | 3 |
| Sets list | 0 | 0 | 0 | 1 | 0 |
| Records (in set) | 0 | 0 | 0 | 1 | 0 |
| Records (in base) | 1 | nrec | nrec | 0 | nrec |

mark — флаг существования описания указанного параметра. Равен 1 если параметр не описан. Например, список выборок имеет mark=1 если предварительно выборка не создавалась инструкциями: find, get или index/create...

### 4.3.4 Дополнительные команды

@/BELL

Включить "звонок" — сигнал ожидания ввода инструкции. Эта же инструкция выключает звонок.

@/EDITOR,<FILEname>,<EDITORsname>

выход из BDMS для редактирования файла <FILEname> редактором EDITORsname. По окончании работы с редактором управление снова переходит к BDMS. (Подробнее в главе 5.)

@/FDT

печатает на экран FDT.

@/HELPBASE

печатает на экран дисплея вспомогательные описаний к текущей базе данных.

@/LS

печатает на экран дисплея список выборок пользователя. (аналог команды DIRECTORY подсистемы EXECUTIVE. См. Гл. 5)

@/SPAWN,"VMS command"

выход в систему VMS для исполнения одной команды пользователя.

@/TTYDRIVER

инструкция на отключение/включение драйвера для дисплея VT52.

### 4.3.5 Параметры вывода

При выводе текста на экран, копировании из файла в файл, допускается управление с помощью параметров:

| | | |
|---|---|---|
| NU | – | пронумеровать строки; |
| FR<n> | – | печатать со строки номер n; |
| TO<n> | – | печатать до строки номер n; |
| LI<n> | – | печатать n строк; |
| FR/<text>/ | – | печатать со строки начинающейся с "text"; |
| FRC/<text>/ | – | печатать со строки содержащей "text"; |
| FRS/<text>/ | – | печатать со строки первые непробельные символы которой: "text"; |

REP — повторить исполнение всех параметров управления выводом до конца файла.

Аналогичные параметрам FR, FRC, FRS параметры TO, TOC, TOS — ограничивают вывод текста по тем же условиям.

Параметры вывода используются в инструкциях: INPUT, INDEX, LOOK, @/LF.

# Глава 5

# Подсистемы BDMS/4

В этой главе мы рассмотрим подсистемы BDMS/4. Подсистемой назовем замкнутую часть системы, выполняющую логически замкнутую часть функций системы без обращения к другим частям системы. Подсистемы в BDMS могут отличаться не только функциями, но и языком. При этом речь не о естественном различии в наборах инструкций, а о более глубоких языковых различиях.

Мы уже рассмотрели некоторые из подсистем BDMS: РЕДАКТОР (EDIT), ПОИСКОВУЮ систему (QUERY), ИНИЦИИРУЮЩУЮ систему (DEFINE), ЗАГРУЗЧИК (LOAD), подсистему вспомогательных (общих) команд. Сюда выведено описание подсистем, подробного рассмотрение которых в контексте общего ознакомления с BDMS структурами и языком не было, (что не является свидетельством маловажности приводимых ниже подсистем для системы в целом!)

Подсистема EXECUTIVE — основная для общения с базой данных. Из этой подсистемы может быть осуществлен переход в РЕДАКТОР, поисковую систему, в систему вспомогательных описаний и т.д.

Подсистема GUIDE — вспомогательная подсистема кодирования данных. Это один из первых шагов, призванных к установлению большего дружелюбия системы по отношению к пользователю-индексатору.

Вспомогательные подсистемы BALANCE, CLEAN, LOAD, DUMP, используются менее активно, но иногда являются необходимыми для административного контроля за качеством и размерами больших PPDS баз данных.

## 5.1 EXECUTIVE

Подсистема EXECUTIVE — самостоятельная процедура предоставляющая пользователю доступ к существующей базе данных. Инструкции подсистемы позволяют добавить новый документ в базу данных, найти записи, удовлетворяющие указанным критериям, и, вслед за тем, распечатать, отредактировать или вычеркнуть эти записи из базы данных. Часть инструкций позволяют пользователю контролировать некоторые аспекты деятельности системы, такие как операции ввода/вывода.

Основной список команд EXECUTIVE дан в виде глаголов английского языка. (Неосновной — список синонимов к основным командам, макрокоманд)

В интерактивном режиме пользователь предупреждается что EXECUTIVE ждет его инструкций следующим титулом:

{имя базы}:READ:EXEC>

При использование подкоманды интерпретатора "?", а так же при наличии ошибки в имени команды подсистемы в интерактив- ном режиме, на экран дисплея будет распечатана следующая таблица:

| BDMS>Level:EXEC$ List of commands: | | | | |
|---|---|---|---|---|
| Add | Append | Audit | Bell | Commands |
| Delete | Directory | Display | Document | Dump |
| Editor | FDT | Find | Get | Guide |
| HelpBase | Host | If | IndexFile | Input |
| LABEL | LD | Line | List | LookFile |
| Merge | Message | ModDocument | Mode | Modify |
| NoAudit | Precision | Print | Purge | Regent |
| Release | Rename | Set | Significance | SortOutput |
| Stop | Suppress | System | Title | User |

ADD - добавить новый документ в базу данных;
APPEND - приписать уже существующий файл для дозаписи;
AUDIT - включить AUDIT;
BELL - включить/выключить звонок;
COMMANDS - инструкция администратора BDMS;
DIRECTORY - распечатать список созданных выборок;
DISPLAY - пометить ЭД разрешенные к документированию;
DOCUMENT - вызвать пользовательскую процедуру вывода;
DELETE - вычеркнуть запись;
DUMP - вывод записи в EXPORT/IMPORT формате;
EDITOR - передать управление внешнему редактору;
FDT - распечатать на экране FDT.
FIND - вызвать подсистему QUERY;
GET - выбрать из базы запись по ее REC-ID;
GUIDE - вызвать процедуру сопровождения кодирования;
HELPBASE - вызвать описание базы данных;
HOST - временный выход в VMS;
IF - контроль числа записей в текущей выборке;
INDEXFILE - вывод списка значений ключевого ЭД;
INPUT - управляемый ввод данных;
LABEL - пустышка. Действий не производит;
LD - вывод списка файлов;
LINE - установить размер входной/выходной строки;
LIST - документирование записи в LOG канал;
LOOKFILES - вывод на экран содержимого файла;
MERGE - ввод из указанного файла;
MESSAGE - сообщение на экран;

MODDOCUMENT - формат вывода для инструкции DOCUMENT;
MODE - формат вывода для инструкции DUMP;
MODIFY - вызов подсистемы РЕДАКТОР;
NOAUDIT - выключить AUDIT;
PRECISION - число десятичных знаков в выводе действительной величины;
PRINT - документирование записей в файл *.PRN;
PURGE - очистить список выборок;
RELEASE - заменить выходной файл;
RENAME - переименовать файл;
SET - сделать ранее созданную выборку "текущей";
SIGNIFICANCE - число значащих цифр при выводе действительной величины;
SORTOUTPUT - упорядочить текущую выборку;
STOP - закончить исполнение программы;
SUPPRESS - пометить ЭД запрещенные к документированию;
SYSTEM - вспомогательная инструкция администратора;
TITLE - вывод титульного текста;
USER - вызов прикладного процессора;

Модификации BDMS в рамках BDMS/4 могут иметь в списке команд инструкции не приведенные в этом тексте. Это предварительные варианты инструкций, действия не производят. Запрос их описания приведет к печати: "NOT YET INMPLEMENTED."

Ниже приведены более подробные пояснения инструкциям под- системы EXECUTIVE.

**ADD**

Инструкция сообщает системе, что необходимо добавить в базу один или несколько документов. При этом система заготовит пустую запись, приписав ей следующий номер идентификатора <REC-ID>. После чего контроль будет передан подсистеме РЕДАКТОР до тех пор, пока пользователь не завершит ввод данных. Число добавленных документов сообщается пользователю по концу работы инструкции:

{n} RECORD(s) ADDED

Если при записи одного документа необходимо пометить его идентификатором rec-id, и этот идентификатор свободен, то используется инструкция:

**ADD,{rec-id}**

В таком формате инструкция ADD используется автоматически при вызове внешнего редактора для редактировании записи системы.

**APPEND**

Приписать к выходному каналу системы уже существующий файл для дозаписи в него. Формат инструкции:

APPEND,{имя канала},{имя файла}

{имя канала} — описывает один из стандартных выходов системы: AUDIT, LOG, DOCUMENT, DUMP, PRINT. По умолчанию к этим каналам системой приписываются файлы: *.AUD, *.LOG (при неинтерактивном режиме), *.DOC, *.DUM, *.PRN.

К каналу AUDIT система приписывает файл *.AUD в моде дозаписи, если он есть в директории пользователя. При этом вводится вспомогательная строка "MESSAGE,{дата старта задания}".

**AUDIT**

Команда "включает" дублирование входных данных в файл имя базы.AUD (если эта возможность была предварительно выключена командой NOAUDIT). Все входные строки из файла или с терминала заносятся в аудит-файл. При нештатном завершении работы системы аудит-файл позволяет восстановить ситуацию.

**NOAUDIT**

Команда "выключает" вывод информации в аудит-файл. Обычно есть смысл использовать команду при неинтерактивной работе с системой, когда не требуется создавать копии входной информации.

**BELL**

Включает и выключает звонок, звучащий при каждом запросе системы к пользователю, т.е., при печати системой титульной информации в интерактивном режиме. По умолчанию звонок отключен.

**DIRECTORY**

Распечатать на экран дисплея список выборок пользователя. При интерактивной работе с системой команда выполняется автоматически при формировании каждой новой выборки.

Инструкция:

DIRECTORY/BACK

приводит к печати списка выборок в обратной последовательности. Команда DIRECTORY дублируется командой @/LS

**DISPLAY,<DEname 1>,...,<DEname 2>.**

Команда открывает для печати командами LIST, PRINT, DUMP ЭД ранее закрытые командой SUPPRESS. Инструкция:

DISPLAY,ALL-DE.

открывает для печати все DE.

Инструкция:

DISPLAY/?

приводит к печати системой на экране дисплея всех ЭД с указанием их статуса, т.е. отдельно открытые для печати и закрытые. (см. также команду SUPPRESS).

**DOCUMENT**

Обращение к прикладному процессору форматирования выдачи. Параметры данной команды зависят от конкретной база данных. параметры команды DOCUMENT должны быть приведены во вспомогательном описании к конкретной базе данных, и доступны по инструкции HELPBASE.

**DELETE,<n>**

По этой команде система вычеркивает из базы данных документы n из текущей выборки. n - порядковые номера документов в текущей выборке. При вычеркивании документов из базы данных производится редактирование индексного файла. При удовлетворительном завершении операции вычеркивания документа система печатает сообщение:

Record <REC-ID> Deleted.

для каждого вычеркнутого из базы данных документа.

ПРИМЕР:

FIND REC-ID<11;**

DELETE,1:10

Record 1 Deleted.

Record 2 Deleted.

....

Record 10 Deleted.

**DUMP,<n>,<ЭД1>...<ЭДi>.**

Команда подобна PRINT, но документы сбрасываются в файл {имя базы}.DUM в формате воспринимаемом редактором системы. Этот файл можно использовать для последующей загрузки базы данных. Введены четыре различных формата печати по команде DUMP, устанавливаемые командой MODE.

**EDITOR,{объект},{имя редактора},{флаг}**

Использование внешней по отношению к системе процедуры редактирования. {имя редактора} — VMS имя редактирующей программы. Например WYLBUR. {объект} — объект редактирования. Если {объект} число — редактируется соответствующая запись в предварительно созданной выборке. Запись при этом документируется в файле EDITOR.MRG. По концу редакции управление вновь передается системе. Если редактировался документ базы данных, то в начало файла EDITOR.MRG приписывается инструкция ADD,{n}, где {n} — идентификатор отредактированной записи, после чего входной канал переключается на ввод данных из файла EDITOR.MRG.

В качестве {объекта} можно указать имя файла.

Параметр {флаг} используется только при редактировании записей. {флаг} — один из символов "N", "A", "L".

| | | |
|---|---|---|
| флаг="N" | – | разультат радактирования будет размещен в новую запись базы данных. Старая при этом сохранится; |
| ="A" | – | будет произведено редактирование сразу нескольких записей базы данных; |
| ="L" | – | открыть LOG файл, который при работе данной инструкции по умолчанию закрыт. |

Параметры {флаг} и {имя редактора} не обязательны. В качестве имени внешнего редактора по умолчанию используется CONFED — который может быть переопределен при инициировании PPDS редактированием файла BDMS$COM:BDMSINI.COM (см. 1.5).

Если выборка содержит только одну запись, то можно не указывать и номер записи: {объект}.

**FDT/{x}**

Вывод на экран дисплея таблицы FDT (File Definition Table) с описанием атрибутов всех элементов данных. Если параметр {x} отсутствует, то таблица будет представлена в следующем формате (каждый ЭД описан одной строкой):

K <l> :proc: <t> <leng> <exi> NAME <<PAR>> (<SYN>)

где

K – метка ключевого элемента;

l – флаг описания ограничений на вывод ЭД вводимых инструкцией SUPPRESS и отменяемых инструкцией DISPLAY (подавлен l="S" или разрешен: l="D");

proc – список процессоров контроля ЭД. IPROC = "I", OPROC = "O", KPROC = "K", QPROC = "Q". Метка каждого процессора сопроваждается описанием процедуры, связывающей представление данных в процессоре с представлением в ядре системы: "c" — convert, "i" — internal, "e" — external. (см. 1.3)

t – тип представления значения ЭД. Для виртуальных ЭД перед t ставится знак "-". Тип кодируется одним символом: "I" — целочисленное значение, "R" — действительное, "C" — текстовое, "N" — чистый узел.

leng – анализируемая длина значения КЭД в индексном списке;

exi – метка наличия КЭД в индексном списке. Если отсутствует = "(-)". Для неключевых ЭД exi="...".

NAME – имя ЭД. Позиция, с которой печатается имя ЭД определена его местом в иерархической структуре записи. Имя элемента второго уровня печатается со сдвигом на три позиции по отношению к ЭД уровня записи, третьего — на шесть, и т.д.

<PAR> — "родитель" данного ЭД

(SYN) — синоним к имени ЭД

Пример FDT таблицы приведен на Рис.3. Здесь используется часть рисунка таблицы, для иллюстрации действия различных значений параметра {x}. Без параметра:

```
  . . .
K  :Ii..Ki..: C  4        SC (shortcode)
K  :Ie......: C  5        R  (reference)
K  :Ie......: C  1          TY  <r>
K  :Ie......: C  1          D   <r>
   :........: N           AUTHORS (auth)
  . . .
```

Допустимы следующие значения параметра {x}:

"P" – команда печатает в файл DUMP.FDT описание структуры базы данных. Этим файлом можно воспользоваться в моде DEFI для создания новой базы данных с такой же структурой;

"N" – печатается краткий список ЭД, (только имя и "родитель"):

```
  . . .
     SC ;    R ;      TY <r>; D <r>
     AUTHORS ;        A <authors> ;
  . . .
```

"C" – печатаются имена ЭД, графически отображаются родственные связи между ЭД;

```
  . . .
  SC
  R
```

```
    TY <r>
    D <r>
  AUTHORS
    A <authors>
  . . .
```

| | | |
|---|---|---|
| "S" | – | печатаются атрибуты только тех ЭД, для которых с помощью команды SUPPRESS указан запрет на печать стандартными системными средствами вывода; |
| "D" | – | печатаются атрибуты только тех элементов данных, для которых не установлен запрет на печать стандартными системными средствами, или снят с помощью команды DISPLAY; |
| "K" | – | печатаются только атрибуты ключевых элементов данных; |
| ?{имя ЭД} | – | на экран дисплея выводится вспомогательное описание данного ЭД, если оно введено администратором базы данных в файл имя базы.HLP. Например, для ЭД SC базы данных DOCUMENTS:<br>FDT/?SC |

```
SC - short code. This is  the unique identifier  for a record.
     SC  is  constructed  automatically by  the system n from
     last author's name, the last two digits  of yearn  and a
     letter. Letter is appended if necessary for n uniqueness.
For example:
SC=ANTIPOV 78D;
```

Если для указанного в инструкции ЭД администратором не заведено вспомогательное описание, то система комментирует:

FDT/?SP

NO INFO ON: "FDTSP"

Команда FDT дублируется командой @/FDT подсистемы вспомогательных команд.

**FIND**

Команда вызывает подсистему обработки запроса пользователя, (подробно рассмотренную во второй главе) выход из которой производится по команде "**". Система просмотрит документы в базе данных и из отвечающих запросу создаст новую текущую выборку записей. Пользователю сообщается номер выборки и количество записей в ней.

<n> RECORD(s) IN SET <s>

<n> может принимать также значение 0.

GET,{номер записи}

Инструкция создает выборку, содержащую указанную в параметре запись, если она есть в базе данных (в противном случае создается пустая выборка).

**GUIDE**

вызвать подсистему сопровождения кодирования. См. следующий раздел этой главы (5.2)

**HELPBASE**

Печатает на экран дисплея вспомогательные описания к используемой базе данных. Описания размещены в текстовом файле BDMS$HELP:имя базы.HLP. Содержимое этого файла определяется администратором базы данных. Формат *.HLP файлов описан в Приложении HELP.

**HOST,"команда VMS"**

Выход в систему VMS для исполнения одной команды пользователя. По исполнении команды управление вновь передается BDMS. В подсистеме вспомогательных команд используется аналогичная по действию @/SPAWN,"команда VMS".

**IF,NREC.{оп}.{n}:{mode}:{действие}**

проконтролировать число записей текущей выборки.

| | | |
|---|---|---|
| NREC | – | имя, зарезервированное за переменной, значение которой равно числу записей в текущей выборке. |
| {оп} | – | логическая операция сравнения. Используется формат, принятый в Фортране. OP= EQ, NE, LE, LT, GE, GT. |
| {mode} = GOTO | – | если результат логического сравнения ИСТИНА, то пропустить все последующие команды и данные вплоть до появления текста отмеченного в параметре "ACTION" = USE — если результат логического сравнения "ИСТИНА" то исполнить "ACTION". |
| {n} | – | целое число, критерий сравнения. |
| {действие} | – | команда, список команд, выполняемых последовательно или макрокоманда или метка (цепочка символов), до появления которой во входной строке все остальные символы будут игнорироваться в случае исполнения заказанного условия. Инструкция LABEL подсистемы EXECUTIVE имеет смысл как метка в потоке команд и данных при использовании инструкции IF. |

Пример использования инструкции:

```
...
IF,NREC.EQ.0:GOTO,LABEL
DUMP PURGE
LABEL
...
```

**INDEXFILE,{имя ключевого ЭД}/{квалификаторы}**

Инструкция позволяет просмотреть упорядоченный список ключевых элементов данных, встречающихся в выборке. На экран дисплея или в файл печатается список значений указанного ключевого ЭД и количество записей в которых они встречаются.

Исполнение инструкции в интерактивном режиме может быть прервано пользователем с помощью STOPLIST процедуры.

/LIST={имя файла} — Вывод списка значений ключевого ЭД в файл: {имя файла}. По умолчанию — INDEX.DAT. При отсутствии квалификатора /LIST - печать производится в LOG канал.

/SUBMIT={имя очереди} — Переход в фоновое задание.

/SET:{n} — печать списка КЭД присутствующих в документах выборки n.

/IF:NREC.{оп}.{n}:{mode} — позволяет выбрать из списка значений KDE только те, чья частота появления в базе данных соответствует указанному в квалификаторе условию.

Формат описания условия аналогичен формату инструкции IF (см. выше). mode - имеет те же значения USE, GOTO, что в команде IF, и означает "печатать" для USE, "пропустить" для GOTO. По умолчанию mode=USE. Квалификатор IF в инструкции INDEXFILE допускает некоторые форматные вольности. Можно опустить NREC:

/IF.<OP>.<Q>

Отсутствие описания эквивалентно инструкции:

/IF:NREC.GT.1:USE == /IF

/CONTAIN:{условие} — Выбор по содержимому значения KDE. Описание условий выбора тождественно принятому при использовании поиска по содержимому КЭД в оформлении запроса (см. вторую главу). Инструкцией INDEX с этим параметром можно использовать для контроля правильности его формирования, прежде чем вводить его в запрос.

/CREATE — создание выборки из записей, содержащих просматриваемые значения КЭД. Необходимо помнить, что при использовании этого параметра будет создана выборка, в которой записи не упорядоченны по номеру (как во всех остальных случаях), и, кроме того, в выборке, полученной таким образом отдельная запись может встретится больше чем один раз.

/LPARAMS:"параметры" — управление выводом списка на экран. Список используемых параметров тождественен списку параметров управления выводом, описанном в главе 4).

/COLUMN:{n} — число колонок для печати списка. По концу работы команды INDEXFILE на экране печатается сводная информация по <KDEname> списку (или его части, при прерывании работы команды пользователем):

```
Total            number of KDE value:  <n1> for <m1> namesn
Selected by set  number of KDE value:  <n2> for <m2> namesn
Displaied        number of KDE value:  <n3> for <m3> namesn
```

**LINE,<l>**

Команда устанавливает размер входной/выходной строки. По умолчанию l=80 позиций.

Система контролирует значение параметра <n>. Минимальное допустимое — 10, максимальное — 132. При выходе за пределы устанавливается значение <n> по умолчанию: 80. Система в таком случае предупреждает:

```
Warning. Invalid numerical value: $<$n$>$ for LINE LENGTH
Used for control: MIN=10, MAX=132. Setted as default: 80
```

**LIST,{номер записи},{ЭД1}...{ЭДi}.**

Команда документирует указанные в числовом параметре номер записи записи выборки в системный файл LOG (при интерактивной работе — на экран дисплея). Если n опущен, то документируются все записи выборки.

Описание каждого ЭД начинается с новой строки. Зависимые ЭД будут печататься следом за тем родительским ЭД к которому они присоединены. Сдвиг от начала строки отмечает уровень подчиненности данного ЭД.

Если в параметрах команды отсутствует список ЭД, контроль списка выводимых ЭД осуществляется по значению заданному предыдущими командами SUPPRESS и DISPLAY. Если эти команды не использовались, то печататься будут все ЭД (исключая виртуальные).

При наличии в параметрах команды списка ЭД разделенных запятыми и завершаемого точкой, печататься будут только они.

**LOOKFILES**

Инструкция используется для вывода на экран дисплея содержимого текстового файла присоединенного в данный момент к одному из каналов ввода вывода BDMS, или же внешнего по отношению к системе. Формат:

LOOK[,имя][,параметры вывода]

где "имя" — либо имя канала ввода/вывода (AUDIT, DOCUMENT, DUMP, PRINT, LOG), либо внешнего файла. В последнем случае необходимо в имени использовать символ "точка", даже если к имени внешнего файла нет расширения.

Формат вывода содержимого указанного в инструкции LOOK файла на экран дисплея определяется "параметрами вывода". В параметрах можно указать с какой по какую строку (по номеру или содержимому строки) производить печать, нумеровать или нет строки, и т.д. (подробнее в 4.2).

Инструкция дублируется аналогичной в "СЛУЖЕБНОЙ подсистеме": @/LF

**MESSAGE**

Message to screen. Может использоваться при оформлении макрокоманд или переключении каналов ввода данных. Формат:

MESSAGE,"<TEXT>"

или

MESSGE,<TEXT>.

**MODE,<m>**

В системный файл имя базы.DUM по команде DUMP текущая выборка может заноситься в разных форматах. Чаще используется так называемый "плотный", при котором документ целиком записывается как непрерывный поток символов, произвольно поделенный на строки равной длины (определенной в параметре команды LINE, или равной 80 по умолчанию).

Другой формат вывода — "редкий". Каждый новый ЭД печатается с новой строки. В конце документа печатается *E. Новый документ всегда печатается с новой строки.

- m = 1 – Редкий сброс. Отмечается иерархия структуры записи, сдвигом начала строки зависимого элемента данных по отношению к родителю. Если сдвиг отмечающий иерархию превышает LINE, то печать производится без сдвига. Если присутствуют несколько одинаковых ЭД, то печатается номер генерации каждого. Печатается идентификатор документа (REC-ID");
- m = 2 – то же что m=1 но иерархия не размечается;
- m = 3 – "плотный" формат (установлен по умолчанию);
- m = 4 – "редкий" формат.

ОТМЕТИМ: обменными являются m=3 и m=4.

**MODIFY,{n}**

По этой команде указанный в параметре документ n переносится из базы данных в память программы, после чего контроль передается редактору системы BDMS. Если редактирование записи удовлетворительно, контроль возвращается в подсистему EXECUTIVE документ записывается в базу данных, все КЭД, встречающиеся в этом документе размещаются в индексном файле. При удовлетворительном исполнении этих операций система печатает сообщение:

Record <n> Modified.

**PRECISION,{p}**

Команда устанавливает число цифр после десятичной точки печатаемых для значения действительной величины. По умолчанию p = 5.

**PRINT,{n},{ЭД1}...{ЭДi}.**

Инструкция идентична LIST, только документы печатаются в системный файл имя базы.PRN, который может быть выведен на печатающее устройство после окончания работы задания.

**PURGE,{номер выборки}**

команда очищает рабочую дисковую память от ранее созданных выборок. Если <номер выборки> отсутствует, вычищаются все выборки, если присутствует, то ТОЛЬКО последняя в списке, вне зависимости от указанного в параметре команды. Если <номер выборки> не последний в списке выборок, то она становится текущей. {Номер выборки} — числовой параметр (см. 5.1.1).

**SET,{номер выборки}**

Команда делает ранее созданную выборку номер выборки текущей, то есть доступной командам LIST, PRINT, DUMP, MODIFY, DELETE. Если числовой параметр номер выборки > 0, то используется указанная выборка, если < 0, то используется {номер последней выборки}-{номер выборки}+1. Номер выборки=-1 означает последнюю в списке выборок. (См. 5.1.1)

**SIGNIFICANCE,{s}**

Команда изменяет параметр s определяющий число значащих цифр, используемых при печати действительного числа. По умолчанию s=5.

**SORTOUTPUT,<DE1>,...,<DEn>.**

Команда используется для упорядочивания текущей выборки пользователя по значению указанных в параметре DE.

Список DEi дает иерархию при сортировке.

Выборка, сформированная по команде SortOutput используется в дальнейшем также, как и любая другая в BDMS. Ее можно сравнивать с другими выборками, выводить на печать, записи в ней можно редактировать.

В списке выборок отсортированная будет представлена записью:

```
   SET  N.REC  COMMAND:
    m   <n1>  :FIND <QUERY>;**
    .
    .
*   l   <n1>  :SORTOUTPUT,<DE1>,...,<DEn>. <sorted set<m>>
```

Сортировать можно по любому ЭД (в том числе и не ключевому). Сортировка алфавитная.

**STOP**

В общем случае команда окончания работы системы BDMS.

В версии BDMS/4.0 введена возможность передавать управление на другой канал ввода данных (команды MERGE или @<file-name>). Команда STOP, или эквивалентный по действию сигнал "конец входного файла" возвращают управление к ранее использовавшемуся каналу ввода.

Сигналом "конец входного файла" при интерактивной работе является символ "^Z".

**SUPPRESS,<DEname 1>,...,<DEname 2>.**

Команда закрывает для вывода командами LIST, PRINT, DUMP перечисленные DE.

Инструкция:

SUPPRESS,ALL-DE.

закрывает для печати все DE.

Инструкция:

SUPPRESS/?

Приводит к печати системой на экране дисплея всех DE с указанием их статуса, т.е. отдельно открытые для печати и закрытые.

**TITLE {текст}**

Записать титульный текст пользователя в файл {имя базы}.DOC. Запись производится шрифтом большого размера, приспособленного к выводу на АЦПУ. {текст} может содержать латинские, русские и греческие символы, а также некоторый набор специальных символов. Для переключения шрифтов используются последовательности символов:

```
 Печатаемые символы (шрифты)      Код шрифта

@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_       #  (по умолчанию)
`abcdefghijklmnopqrstuvwxyz{|}~

юабцдефгхийклмнопярстужвьызшэщчъ       $
ЮАБЦДЕФГХИЙКЛМНОПЯРСТУЖВЬЫЗШЭЩЧ

. . .
```

Например, инструкция:

TITLE, $pROWERKA INSTRUKCII #TITLE. &abgd.

приведет к выводу в файл {имя базы}.DOC текста:

Проверка инструкции

title. $\alpha\beta\gamma\delta$.

Если использованный в параметре текст имеет пробелы и по размерам не помещается в строку DOC файла, то он будет по пробельным позициям разделен на несколько строк.

По концу работы инструкции TITLE восстанавливается латинский шрифт, если были использованы переключатели шрифта "$", "&", "?". Если символы переключения шрифта требуется использовать в тексте, то их нужно печатать более одного раза.

**USER**

Этой инструкцией вызывается прикладная процедура, если она предусмотрена администратором базы данных.

### 5.1.1 Числовые параметры к командам EXECUTIVE

Некоторые команды подсистемы executive имеют числовой параметр, указывающий либо на номер записи в выборке (MODIFY, DUMP, LIST, PRINT, EDITOR), либо номер записи в базе данных (GET, FIND), либо на номер выборки в списке выборок (SET, SORTOUTPUT, FIND). При этом, если параметров может быть несколько, то первым ставится числовой.

Числовые параметры есть и в других командах, например LINE, MODE, PRECISION, SIGNIFICANCE, но здесь мы рассмотрим только первые три перечисленные выше группы.

Числовой параметр — это группа чисел, как положительных, так и отрицательных, разделенных запятой или соединенных двоеточием. Например, инструкции:

| | | |
|---|---|---|
| LIST,1,13,22,15,5 | – | документы выводятся на экран в том порядке, в котором они перечислены в списке; |
| DUMP,8:15 | – | вывести в файл имя базы.DUM из текущей выборки записи с 8 по 15; |
| DELETE,-1:-4,6 | – | удалить из базы собранные в текущей выборке записи: последние четыре в выборке и шестую, (при этом не анализируется, входит ли шестая запись в последние четыре); |
| SET,-2 | – | делает текущей вторую с конца выборку, из списка выборок; |
| FIND (3) AND (-1)** | – | пересечение третьей и последней выборки. |

Значения числовых параметров контролируется специальной процедурой, в которой заданы для данного параметра область его определения и значение принимаемое по умолчанию. Список контролируемых параметров и их значения можно запросить у системы инструкцией @/LP.

### 5.1.2 STOPLIST процедура

Если пользователь заказал вывод какого-либо текста на экран дисплея, то система контролирует число передаваемых строк, прерывая печать после некоторого их количества (страницу). При этом пользователь может либо продолжить вывод, либо прервать.

Программа приглашает пользователя к диалогу запросом:

#L= <l> : <m> More? ->

где

<l> — число уже отпечатанных строк;

<m> — длина страницы.

Реакция пользователя на это приглашение определяется следующим набором команд:

```
Stop (или End или Finish или K,X,H,N ^Z)
+                        <m> = <m>*2
-                        <m> = <m>/2
<digit> from 1 to 240 -- установить размер страницы
                         <m>=<digit>
*   печатать без прерывания
M - установить максимальную длину страницы (240)
J - перейти на следующую страницу не печатая текущую
L - напечатать только одну строку
пустая строка - продолжить печатать
? - печать на экран сообщения о перечисленных выше командах.
```

## 5.2 Автокодировщик (GUIDE подсистема)

Автокодировщик создан для решения следующих задач:

1. освободить кодировщика от необходимости контролировать правильность написания имен ЭД, от необходимости следить за соответствием данных описанию структуры документа;
2. предоставить кодировщику своевременно напоминание о правилах кодирования (индексирования);
3. производить контроль качества непосредственно при кодировании;
4. расширить возможности контроля кодирования;
5. упростить процедуру кодирования;
6. предоставить достаточно простую процедуру управления самим автокодировщиком.

Информация о структуре документа поступает в автокодировщик непосредственно из FDT. На основании этой информации автокодировщик выбирает ЭД для кодирования и контролирует корректность конструирования структуры кодируемого документа. Такой подход позволяет устранить возможность ошибок в именах ЭД, и опасность несоответствия структуре документа. (В автокодировщике отсутствует ввод сразу группы значений одного и того же ЭД.)

По умолчанию автокодировщик предлагает заполнять последовательно значения ЭД определенных в FDT. ЭД, не имеющие зависимых элементов, кодируются в цикле до тех пор пока кодировщик не откажется от заполнения очередной генерации данного ЭД.

Правила заполнения значений ЭД описаны в модели кодирования, которая может быть системной (создана администратором) или пользовательской.

### 5.2.1 Команды автокодировщика

Автокодировщик обращается к пользователю с запросом на кодирование ЭД:

{имя ЭД} [краткое пояснение]=

Реакция пользователя — введение значения ЭД в соответствии с форматом, который может быть продемонстрирован в "кратком пояснении" или в комментарии к ЭД.

Символ ";" означает, что пользователь завершил заполнение значения ЭД.

При каждом нажатии клавиши RETURN первые символы анализируются автокодировщиком как возможные команды процедуры "GUIDE".

В настоящее время введены следующие команды автокодировщика:

| | | |
|---|---|---|
| | – | (пустая строка): не заполнять значение текущего ЭД; |
| ? | – | распечатать на экране описание возможных действий кодировщика в текущей ситуации; |
| ? | – | распечатать на экране описание ЭД из текста к параметру COMMENT модели кодирования, если он существует, или из вспомогательного описания к базе данных; |
| L | – | распечатать на экране уже закодированную часть документа; |
| G | – | перейти в процедуру управления моделью кодирования; |
| C | – | прервать кодирование без возможности восстановления; |
| B | – | прервать кодирование, вернуться на уровень управления с возможностью возвращения в автокодировщик и восстановления текущей ситуации. Восстановления текущей ситуации в автокодировщике не произойдет, если после команды *B производился выход из системы. |
| M | – | вызов процедуры редактирования уже закодированной части документа; |
| *D | – | вычеркнуть уже закодированную, но еще не закрытую символом ";" часть значения ЭД; |
| P | – | "primer", вызов процедуры INDEX (см. 9.5) для текущего ЭД; |
| F | – | Вызов процедуры FDT (см. 9.3) |

Если база данных снабжена входными процессорами контроля, то соответствующее значение ЭД будет проконтролировано и информация об обнаруженных ошибках высвечена на экран. Если ошибка кодирования признается процессором фатальной, то автокодировщик тут же предложит заменить или исправить ошибочное значение.

Ошибка в уже закрытом символом ";" значении ЭД может быть исправлена до завершения кодирования документа с помощью встроенного редактора (команда "*M").

По завершении кодирования заполненный документ добавляется в базу данных. При этом процессоры контроля просигнализируют на экран дисплея об ошибках. Теперь уже ошибки придется исправлять с помощью команд MODIFY или EDITOR (см. 9.5)

### 5.2.2 Модель кодирования

В автокодировщике реализована возможность конструировать разнообразные модели кодирования для одной и той же базы данных. Команда автокодировщика *G передает управление процедуре формирования модели кодирования (см. 11.4).

Модель размещается в файле <DBname>.GDN, который читается системой по умолчанию. В этом файле описываются ЭД и список параметров, которыми задается модель кодирования ЭД.

По умолчанию автокодировщик использует <DBname>.GDN из директории пользователя. Если этот файл отсутствует, то файл с таким-же именем из системной директории.

Если нет внешних описаний правил кодирования — автокодировщик будет использовать "пустой" FDT, как модель кодирования. В любом случае автокодировщик представляет возможность создавать, менять и выбирать модель кодирования в процессе ввода данных.

В модели каждому ЭД приписан набор параметров кодирования. В настоящее время это:

**MAIN**

ЭД, отмеченный этим флагом, рассматривается как необходимый для кодируемого документа, т.е., без значения этого ЭД документ не может существовать. Попытка кодировщика проигнорировать этот ЭД будет пресекаться программой:

```
Sorry! This data element is marked as MAIN for encoding.
       You must put value for this one.
```

**UNIQUE**

Значение этого ЭД заполняется только раз (в пределах одной ветви структуры).

**IGNORE**

ЭД будет пропущен при кодировании (со всеми зависящими от него).

**BIGLETTERS**

Значение ЭД будет заполнено заглавными буквами вне зависимости от того, какие использовал кодировщик при вводе.

**SPELL**

Контроль орфографии. При обработке введенного текста слова не найденные в словаре системы будут отмечены. Автокодировщик, обнаружив отсутствующие в словаре слова, предложит их заменить.

ПРИМЕР: ЭД TEXT со включенным параметром SPELL. Вводится:

```
<TEXT > = Firt autor name;
          Возможный диалог с пользователем:

BDMS:ENCODER:Spelling control> Wrong words:
   FIRT          AUTOR
Would you want to edit these words? > Y
FIRT <<<< First
First AUTOR <<<< author
```

**DISPLAY:**"краткое пояснение" —

Текст "краткое пояснение" будет печататься на экране вместе с именем ЭД при обращении к пользователю заполнить значение ЭД. "Краткое пояснение" все же несколько более широкое описание ЭД, чем то, что дает одно имя ЭД.

ПРИМЕР: для ЭД "SC"

DISPLAY:"Short Code:"

программа напечатает в запросе:

<SC > Short Code =

**MENU:**"ключ1;ключ2;,...,ключn;" —

Для элементов данных чьи значения определены на достаточно ограниченной области различных ключевых выражений "ключ1, ключ2, ..., ключn.". При наличии этого параметра кодировщик может заполняя значение ЭД либо записать его "ключ i;", либо указать на один из предложенных программой номеров меню.

ПРИМЕР: если для ЭД "TY Publication type" в модели кодирования введен параметр:

MENU:"JOUR;PREP;CONF;PRIV;TOBP;"

то, программа следующим образом предложит заполнить этот элемент:

```
<TY > Publication type =          n

JOUR PREP CONF PRIV TOBP          n
 [1]  [2]  [3]  [4]  [5]  >       n
```

и ответом, если описываемый тип публикации JOUR (журнальная статья) будет либо явное описание значения:

JOUR;

либо просто номер 1.

Наличие параметра MENU в модели кодирования не препятствует обычному вводу значения, если он оказался не охваченным предлагаемым меню.

**MACRO:**"текст a#текст A;...;текст m#текст M;" —

набор символов "текст i" во входной строке должен быть заменен в значении ЭД набором символов "текст I". Если "текст I" не задан, т.е.: "текст i#;" то "текст i" будет просто удален из значения.

ПРИМЕР:

MACRO:"1987#87;"

в значении ЭД всюду "1987" напечатанное пользователем при вводе будет заменено на "87".

**CONSTANT:**"текст" —

"текст" — постоянно используемое сочетание символов в значении кодируемого ЭД. Изменяемая часть сочетания символов отмечается символом "?". Программа потребует заполнить изменяемую часть, самостоятельно разместив постоянную в значении ЭД. Вместо "#D" в текст будет подставлено текущее значение даты.

ПРИМЕР:

CONSTANT:"Encoded #D by ?"

при кодировании ЭД имеющего такое описание параметра CONSTANT программа напечатает на экране:

```
Type needed VALUE INSTEAD of "?"
ENCODED 5 AUG 1987 BY ? <
```

**COMMENT:**"текст" —

"текст" — комментарий. Текст дополнительной информации о ЭД. Длина текста не ограничена. Задача: уточнить особенности заполнения значения данного ЭД, не выявляемые его именем и кратким пояснением к нему. Текст комментария печатается автокодировщиком по команде "*?". При заполнении описания параметра COMMENT символ "#" используется как форматирующий и означает "начало новой строки" при печати текста на экран.

**CALCULATION:**{"выражение"}

используется для перевычисления входного численного значения. Формат описания "выражения":

"{тип}{(количество чисел ЭД)}{,функция}{,}[константа]

{,точность}"

Если в модели указана команда на перевычисление, но параметры к ней отсутствуют, то программа предложит их ввести.

ПРИМЕР:

CALCULATION:"R(3),/,6.32,5"

перевычислить три "(3)" действительных "R" числа в значении ЭД, разделив "/" их на константу "6.32" и результат дать в десятичном представлении с точностью 5 цифр после десятичной точки.

Тип: "I" — целое; "R" — действительное; "R*8" — действительное двойной точности.

В описании параметра используются все фортрановские функции:

```
+, -, *, /, **, SQRT,   ALOG,  ALOG10, EXP,   SIN,   COS,
TAN,    ASIN,   ACOS,   ATAN,  ATAN2,  SINH,  COSH,  TANH,
ABS,    AINT,   ANINT,  AMAX1, AMIN1,  DIM,   AMOD,  SIGN,
DSQRT,  DLOG,   DLOG10, DEXP,  DSIN,   DCOS,  DTAN,  DASIN,
DACOS,  DATAN,  DATAN2, DSINH, DCOSH,  DTANH, DABS,  DINT,
DNINT,  DMAX1,  DMIN1,  DDIM,  DMOD,   DSIGN, IAND,  IOR,
IEOR,   NOT,    ISHFT,  MAX,   MIN,    IDIM,  MOD,   ISIGN.
```

Константа используется только для бинарных операций, например +, IAND, AMIN,... В остальных случаях игнорируется.

### 5.2.3 Установка параметров модели кодирования

Управление параметрами кодирования (редактирование модели сопровождения кодирования) реализовано в самом АВТОКОДИРОВЩИКЕ. По команде автокодировщика "*G" контроль передается интерактивной процедуре редактирования параметров.

Процедура запрашивает команду пользователя титулом:

<имя базы>:DEN=<имя ЭД>:Encod_Initia:>

где "имя ЭД" - имя ЭД который должен кодироваться.

В ответ пользователь может напечатать новое значение параметров модели кодирования, которые рассматриваются интерпретатором команд как команды данной подсистемы:

| | | |
|---|---|---|
| *E | – | или пустая строка — вернуться в автокодировщик; |
| L | – | распечатать параметры модели кодирования текущего ЭД; |
| L,* | – | распечатать параметры модели кодирования всех ЭД; |
| F | – | распечатать FDT; |
| P | – | аналогично команде INDEXFILE для текущего ЭД, если он ключевой; |
| N,<ЭД> | – | сделать указаный ЭД текущим для редактора параметров модели кодирования; |
| S,<FILE> | – | переписать параметры модели кодирования в файл <FILE>. Если <FILE> не задано, по умолчанию создается файл <DBname>.GDN. |
| I,<FILE> | – | прочитать параметры из файла <FILE>. Если параметр <FILE> отсутствует, то по умолчанию из <DBname>.GDN в директории пользователя, и, если нет пользовательского, то используется созданный администратором из системной директории. |

Параметры модели кодирования можно ввести в программу по очереди или одновременно для одного ЭД. В последнем случае параметры отделяются друг от друга символом "/". Если не хватает длины входной строки, можно продолжить задание параметров на другой, завершая предыдущую символом продолжения "-".

Все параметры перечисленные выше можно использовать с приставкой "no". Например: "nomain" — означает отменить действие параметра "main" для указанного ЭД.

Список наименований параметров модели кодирования, как и любой список команд в данной версии BDMS может быть получен при использовании подкоманды "?". Описание параметров выводится на экран по команде:

?{имя_параметра}

### 5.2.4 Редактор автокодировщика

В автокодировщик для внесения исправлений в уже закодированную часть документа, включен мини-редактор с ограниченным числом команд. Этот редактор не позволяет редактировать имена ЭД, вставлять пропущенные или удалять уже вставленные (то есть, вмешиваться в конструирование структуры).

Процедура опрашивает пользователя титулом:

ENCODER:EDIT>{VER}.:{TEST}>

VER. — флаг, что редактирование каждого ЭД будет принято только после подтверждения пользователем. Отменяется и выставляется командой VERIFY.

TEST — флаг, что отредактированное значение будет контролироваться пользовательскими процессорами и, если указано в модели, орфографическим контролем. Отменяется и выставляется командой TEST.

**TRAP,DElist**

Задание списка ЭД, значения которых разрешено редактировать.

**DISPLAY**

Просмотр редактируемого документа.

**PROTECT,DElist**

Команда, защищающая поименованные в списке ЭД от случайного редактирования. По умолчанию все ЭД рассматриваются как запрещенные к редактированию.

Собственно редактирование может быть осуществлено двумя командами:

**CHANGE,DElist oldtext newtext**

команда найдет "oldtext" во всех ЭД разрешенных для редактирования, и, заменит oldtext на "newtext.

**CALCULATE,DElist digit "text"**

команда перевычислит численное значение digit во всех доступных для редактирования значениях ЭД в соответствии с правилами, данными в "text". Правила перевычислений формулируются так же, как и одноименный параметр модели кодирования.

DElist — список ЭД на чьи значения допустимо воздействие команды, расположенной перед этим списком. ЭД в этом списке разделяются запятыми. DElist дополняет список ЭД отмеченных командой TRAP, но ЭД приведенные в этом списке доступны действию только текущей команды редактора.

**SEARCH,DElist {text}**

найти и распечатать на экране все значения ЭД содержащие указанную в параметре "text" группу символов.

**LIST,DElist**

Инструкция для просмотра редактируемого документа.

**FORGET**

В случае неприемлемой редакции ее результат может быть забыт.

**EXIT**

Выход из редактора.

## 5.3 Сброс (DUMP)

Подсистема СБРОС "DUMP" используется для сброса документов базы данных в формате доступном для последующей загрузки в базу данных. Используется если необходимо перенести базу данных на другую ЭВМ. Подсистема достигает максимальной эффективности: записи сбрасываются из файла *.FIL последовательно в случайном порядке, (без использования индексной информации и номеров записи.) Если требуется сбросить только часть базы данных, то рекомендуется использовать команды подсистемы EXECUTIVE: FIND и DUMP.

Подсистема СБРОС читает из входного файла инструкции, специфицирующие формат выходного файла. Это:

**LINE,{n}**

Инструкция устанавливает длину выходной строки равную n символов. По умолчанию длина строки равна 80 символам.

**MODE,{n}**

Выбор формата вывода документов. Для —n—=1 или 2 документ оглавляется номером записи в базе (REC-ID). При —{n}—=3 или 4 формат приспособлен для использования в качестве входного для подсистемы LOAD. По умолчанию установлен параметр {n}=4.

**DUMP**

Эта команда завершает набор входных инструкций и инициирует процесс сброса. Команда DUMP быть указана даже если нет никаких других команд.

Все инструкции отделяются друг от друга пробелом и, поэтому, не могут содержать включенных пробелов. Выход помещается в файл DBname.DUM.

## 5.4 Чистка (CLEAN)

Эта подсистема используется для уплотнения файла *.FIL, чистки пустой памяти возникшей в результате пополнения базы данных. Частота использования подсистемы зависит от активности пополнения и редактирования базы данных.

Подсистема не требует входных файлов.

## 5.5 Балансировка индексных деревьев

Подсистема "BALANCE" используется для перебалансировки инвертированных списков ключевых ЭД. Эта операция используется после больших (сравнимых по объему с базой данных) дополнений, и призвана оптимизировать поиск при отработке запроса пользователя.

Из входного файла подсистема воспринимает следующие команды:

**FREE,{n}**

Установка размера свободного пространства на каждой странице индексного списка для использования при последующих дополнениях в базу данных без возможных разбалансировок индексных деревьев. Если n не равно 0, то n - свободная часть страницы в процентах. Если равно, то страницы будут заполняться полностью для достижения минимума размера базы данных. По умолчанию установлено 50

**BALANCE**

Эта команда завершает набор инструкций системе и включает процедуру ребалансировки.

## 5.6 Загрузка (LOAD)

Подсистема LOAD используется для начальной загрузки базы данных. Ее нельзя использовать для дополнения записи в уже существующую базу данных (для этой цели используется инструкция подсистемы EXECUTIVE ADD. Загрузка данных подсистемой LOAD намного более эффективна, чем дополнение, поскольку входы индексных данных последовательно запасаются пока в базе данных не будут размещены все записи, а затем, специальная процедура по полному списку индексируемых значений начинает построение древесной структуры, максимально соптимизированной к поисковому процессору.

Входным данным (документам), предшествуют одна или более команд подсистемы LOAD. Данные записаны в формате встроенного системного редактора. Документы отделяются друг от друга двухсимвольной командой редактора "*E", последний документ завершается либо концом файла, либо "**".

Команды, предшествующие входным данным:

**LINE,{n}**

Инструкция устанавливает длину выходной строки равную n символов. По умолчанию длина строки равна 80 символам.

**FREE,{n}**

Установка размера свободного пространства на каждой странице индексного списка для использования при последующих дополнениях в базу данных без возможных разбалансировок индексных деревьев. Если $n \neq 0$, то $n$ — свободная часть страницы в процентах. Если равно, то страницы будут заполняться полностью для достижения минимума размера базы данных. По умолчанию установлено 50% заполнение.

**LOAD**

Этой инструкцией завершаются входные команды и начинаются данные. LOAD обязательно должна предшествовать данных даже если не используются остальные команды подсистемы.

При старте системы в неинтерактивном режиме выбор подсистемы осуществляется по наличию ОСНОВНОЙ инструкции подсистемы, (в данном случае LOAD), в первой строке входного файла. Если таковая отсутствует, то по умолчанию выбирается EXECUTIVE подсистема.

Все команды подсистемы отделяются друг от друга пробелом.

# Глава 6

# Сообщения об ошибках

При работе с системой в интерактивном режиме на экран печатается вспомогательная информация, помогающая ориентироваться в текущем контексте, анализировать ошибочную ситуацию и находить из нее выход. Каждый контекст маркируется титулом, при необходимости сопровождаемым кратким пояснением. Например, инструкция MERGE или @имя файла:

MERGE> Assigned file: "{имя файла}"

если указанный файл открыть не удалось (отсутствует, недоступен), то система предупреждает:

MERGE> Sorry! I can't open file "{имя файла}"

MERGE> File name: S to exit

что означает предложение заменить ранее указанное имя или, использовав "S" выйти из контекста "MERGE".

Если при обращении к системе неоднократно повторяются некорректно используемые инструкции (например на вывод документов при отсутствии выборок), то система печатает на экране дисплея краткую информацию о том, как ознакомится с ее возможностями.

Ниже приведены пояснения к сообщениям системы об ошибках и предупреждениям, выводимым на экран дисплея. Сообщения могут поступить как из ядра системы, так и из ее процессоров. В отличии от сообщений ядра системы, сообщения процессоров не имеют фиксированного формата. В данной работе приводится пояснения только к наиболее ответственному процессору, ответственному за контроль PPDL данных, используемому в основных базах данных PPDS.

## 6.1 Сообщения из ядра системы

### 6.1.1 Интерпретатор команд

Во всех подсистемах, снабженных интерпретатором команд, при ошибке в написании команды в интерактивном режиме печатается сообщение:

{error verb} Not a command

а также доступный в используемом контексте список команд. При повторении ошибки список команд контекста уже не печата- ется. (Если же требуется его воспроизвести на экране, можно воспользоваться подкомандой интерпретатора "?". см. 4.1)

Сообщение:

{verb} Ambiguous command.

{command 1} {command 2} ... {command n}

означает наличие неразрешимой для интерпретатора команд неоднозначности, возникшей при использовании сокращенного имени команды. В сообщении приводится список неразличенных команд.

При подстановке синонима к имени команды или описании макрокоманды интерпретатор сообщает:

Not unique new user-dependent name: "{new name}"

Type new name:

если новое имя уже представлено в списке команд контекста, и

{old name} is absent in command list

Type old name

если синоним приписывается команде отсутствующей в списке команд контекста. В обоих случаях от дальнейшего выполнения инструкции можно отказаться введя пустую строку.

### 6.1.2 Подсистема DEFINE. Компилятор

Сообщение об ошибках сопровождается префиксом ***ERROR***, опущенным в описании.

Record buffer empty - - list aborted

использованы команды COMPILE или MODIFY ранее, чем команда DEFINE.

<Modi: Illegal recurrence of record-level data element

ЭД уровня записи (FILE или DE) введен повторно без последующей точки.

INVALID PARENT SPECIFIED FOR DE.{n}

При описании {n}-ого ЭД приведена ссылка на еще не определенного родителя.

INVALID TYPE SPECIFIED FOR DE.{n}

Ошибка в описании значения атрибута TY {n}-ого ЭД. Допустимые значения: INTEGER, REAL, DOUBLE, CHARACTER, BIT, NODE.

INVALID IPROC TYPE SPECIFIED FOR DE.{n}

INVALID OPROC TYPE SPECIFIED FOR DE.{n}

INVALID KPROC TYPE SPECIFIED FOR DE.{n}

INVALID QPROC TYPE SPECIFIED FOR DE.{n}

Ошибка в описании значения атрибутов вызова прикладных процессоров контроля. Допустимые значения EXTERNAL, INTERNAL, CONVERT.

NO DATA ELEMENTS DEFINED

FDT документ не содержит ни одного описания атрибута имени элемента данных (DE).

NO TYPE SPECIFIED FOR DE.{n}

Опущено обязательное описание типа ЭД.

PROC SPECIFIED FOR PURE NODE DE.{n}

Бессмысленно передавать узловой ЭД не имеющий значения в прикладной процессор контроля.

WORK SPACE EXCEEDED - - COMMAND SKIPPED

Размер FDT документа превысил доступную рабочую память.

Подсистема не чувствительна к некоторым возможным логическим ошибкам входного документа. Нет реакции компилятора на:

- пересечение имен ЭД в пределах одного FDT документа;
- ссылки из одного ЭД сразу на несколько описанных ранее;
- использование нескольких типов значений при описании одного ЭД;. То же для других атрибутов ЭД.

При детектировании фатальной ошибки в описании документа FDT компилятор печатает сообщение:

COMPILATION TERMINATED

В диалоговом режиме можно вернуться в редактор и исправить найденные компилирующей программой ошибки (инструкция MODIFY), или использовав инструкцию DEFINE повторить ввод документа FDT с начала.

### 6.1.3 Подсистема EXECUTIVE

There are only {n} records in current set.

Command terminated.

В инструкциях DELETE, MODIFY использованы номера записей от- сутствующих в текущей выборке.

INVALID SET NUMBER

Указан номер выборки больший последнего в списке выборок.

RECORD NOT DELETED

Означает, что указанную запись невозможно вычеркнуть из базы данных.

RECORD NOT FULLY DE-INDEXED

Не удалось перезаписать часть или все индексные значения при вычеркивании или редактировании записи. В результате может возникнуть ошибка при выборе записей по запросу.

RECORD NOT FULLY INDEXED

Не удается ввести в древесную структуру списка индексных значений новое значение при дополнении записи в базу данных или дополнении ключевого ЭД при редактировании существующей записи.

RECORD NOT RETRIEVED

Запись указанную в текущей выборке невозможно восстановить в память из базы данных при использовании команд LIST, PRINT, DUMP, ...

RECORD NOT STORED

Не удалось поместить в базу данных новую или модифицированную запись.

WORK SPACE EXCEEDED

Не хватило рабочей памяти для выполнения какой-либо заданной операции. Для выхода из ситуации потребуется перекомпилировать систему с большей рабочей памятью.

Continue list of DE's for printing.

It must be terminated by "."

В командах вывода или SUPPRESS, DISPLAY не поставлен символ конца списка ЭД: "." После этого сообщения можно либо продолжить список ЭД, либо завершить его символом ".".

Undefined data element name:"error DE name"

List of DATA ELEMENTS names:

.......

For more information use FDT command

Сообщение об ошибке в имени элемента данных в списке имен ЭД в инструкциях вывода. При первой ошибке такого типа системой печатается список имен ЭД и подсказка.

"FETCH ERROR" Deleted record

<EXEC:OUTPUT> record not retrieved

Попытка произвести манипуляции с записью ранее вычеркнутую по инструкции DELETE.

<EXEC:OUTPUT> WARNING: There are no records for printing

Инструкции LIST, DUMP, PRINT используются на "пустой" выборке.

<EXEC:OUTPUT> WARNING: You must create any set previously!

Команды вывода использованы до того, как были созданы выборки.

Sets array is empty.

Попытка модифицировать или стереть запись при отсутствии выборки в памяти системы.

Warning. Invalid numerical value: 0 for RECORDS INTO SET

RECORDS INTO SET parameter is not defined.

или

Used for control: MIN=1, MAX=0, Setted as default: 0

### 6.1.4 Информация об ошибках в запросе

Ниже приведен список сообщений системы об ошибках, которые генерятся при обработке запроса пользователя. Сканирование базы данных по запросу начинается только после того, как оно полностью проверено на отсутствие формальных ошибок.

INCOMPLETE QUERY

Ситуация, при которой входной файл дочитан до конца а описание запроса не завершено. Нет "**".

INVALID LENGTH RETURNED BY QPROC

Прикладной процессор QPROC вернул длину преобразованного значения ЭД меньшую нуля.

INVALID NUMERIC DATA ELEMENT VALUE

В значении описывающем ЭД числового типа нет данных соответствующих данному типу.

INVALID RANGE SPECIFICATION

Бессмысленное описание области выбора при использовании операторов отношения: например X $<$ 7; TO $>$ 9;.

NON-KEY DATA ELEMENT IN QUERY

В запросе использован ЭД не описанный как ключевой.

QUERY SYNTAX

Запрос оформлен с ошибками.

TOO MANY LEFT PARENTHESES

TOO MANY RIGHT PARENTHESES

Не совпадает число правых и левых скобок.

UNDEFINED DATA ELEMENT NAME

Ошибка в имени элемента данных. Оно не распознается при сравнении с FDT списком данной базы данных.

VALUE SPECIFIED FOR PURE NODE DATA ELEMENT

В запросе указаны условия на значение для чисто узлового ЭД, что бессмысленно, поскольку чисто узловой элемент данных не содержит значения.

WORK SPACE EXCEEDED

Система вышла за пределы рабочей памяти при обработке поискового предписания. Для использования системы с такой базой данных потребуется перекомпилировать ее, увеличив размер рабочей памяти.

### 6.1.5 Подсистема РЕДАКТОР

В этом разделе рассматриваются сообщения об ошибках подсистемы РЕДАКТОР.

В этой секции поясняется смысл сообщений об ошибках из подсистемы РЕДАКТОР. Они должны быть ясно поняты, так как РЕДАКТОР не позволяет правильно интерпретировать все ошибки во входных данных, (проблемы возникают при многократном определении значения ЭД.

В общем случае при обнаружении ошибки система пропускает часть данных ее содержащую и продолжает обрабатывать следующие входные данные. В моде дозаписи в таком случае некоторые генерации элементов данных не будут созданы. В редактирующей моде не будут исполнены некоторые редактирующие операции. При интерактивной работе ошибки можно тут же исправлять. В неинтерактивной одна ошибка может иногда вызвать возникновение нескольких других, к примеру, если в результате ошибки не создался родительский элемент данных, то последующие подчиненные генерации элементов данных будут подсоединены к ранее созданной генерации родительского.

DATA ELEMENT VALUE IN DELETE COMMAND

Бессмысленно описывать значение вычеркиваемой генерации элемента данных. За командой *Delete может следовать только имя элемента данных завершаемое точкой с запятой.

ILLEGAL RECURRENCE OF RECORD-LEVEL DATA ELEMENT

Данное сообщение возникает в том случае, если в моде дополнения используется имя элемента данных без точки при модифицировании записи уже имеющей хотя бы одну генерацию этого элемента данных. При дописывании записей в базу данных ситуация не отмечается как ошибочная. Редактор в этом случае автоматически генерирует следующую запись.

INCOMPLETE ASSIGNMENT STATEMENT

Возможно отсутствует закрывающий описание значения элемента данных символ ";", а ввод входных данных в неинтерактивной моде уже завершен. При интерактивном вводе данных система будет ждать продолжение описания значения до появления ";".

INCORRECT QUALIFIER

Ошибка в описании номера генерации.

INVALID LENGTH RETURNED BY IPROC

Подпрограмма прикладного процессора IPROC возвращает отрицательную величину длины значения элемента данных.

INVALID NUMERIC DATA ELEMENT VALUE

Значение элемента данных численного типа не распознается как число.

MISSING DATA ELEMENT VALUE

В случае неузлового элемента данных непосредственно за именем следует точка с запятой. Это допустимо только при использовании команды delete.

За один раз в командах replace и substitute может использоваться только одно значение ЭД.

NO MATCHING SUBSTRING

В просматриваемом тексте нет указанной в команде substitute подцепочки символов.

NON-CHAR DATA ELEMENT IN SUBSTITUTE COMMAND

Команда Substitute может действовать только на элементы данных текстового типа.

PARENT DATA ELEMENT MISSING

Редактирующая операция не может быть выполнена поскольку не указана генерация родительского элемента данных.

PURE NODE DATA ELEMENT IN REPLACE OR SUBSTITUTE COMMAND

Бессмысленно заменять значение узлового ЭД или пытаться редактировать его командами Replace или Substitute.

QUALIFIED DATA ELEMENT NAME IN APPEND MODE

В моде дополнения имя элемента данных не может использоваться с указанием номера генерации.

SPECIFIED OCCURRENCE DOES NOT EXIST

Генерация ЭД описанная в команде редактора не присутствует в базе данных. Возможно некорректно введен номер генерации.

SUBSTITUE COMMAND SYNTAX

Ошибка в синтаксисе команды Substitute, т.е. делящий символ не встречается точно три раза или между последним делящим символом и точкой с запятой стоит не пробел.

UNDEFINED DATA ELEMENT NAME

В потоке входных данных не содержится имя элемента данных которое может быть распознано и данных не могут быть интерпретированы как другое значение ранее введенной генерации элемента данных.

WORK SPACE EXCEEDED

Имя элемента данных или значение больше, чем доступная рабочая память системы или нет доступной памяти для расширения стека. В общем случае, систему нужно перекомпилировать с большей рабочей памятью, чтобы застраховаться от ошибки при работе с данной базой данных. Иногда для решения проблемы бывает достаточно переместить запись в базу данных и снова вернуть ее в редактор. При этом будут убраны "мертвые" зоны в памяти, возникающие при редактировании.

### 6.1.6 Процедура LIST

В данной секции приводятся сообщения об ошибках генерируемые при документировании записи.

INVALID LENGTH RETURNED BY OPROC - - DATA ELEMENT SKIPPED

Из прикладного процессора QPROC возвращена отрицательная величина длины значения элемента данных.

INVALID MODE PARAMETER - - LIST ABORTED

Подобное сообщение возможно только при использовании команды LIST из прикладного процессора, если вызов системной функции LIST произведен с неправильным описанием параметра MODE.

RECORD BUFFER EMPTY - - LIST ABORTET

Запись, которую требуется распечатать, отсутствует в памяти. Ситуация возникает если пользователь использует команду LIST непосредственно, без предварительного инициирования записи в буфере программы.

WORK SPACE EXCEEDED - - DATA ELEMENT SKIPPED

WORK SPACE EXCEEDED - - LIST ABORTED

Размер записи не позволяет разместить ее в рабочей памяти. Требуется перекомпилировать систему с большей рабочей памятью для работы с такой базой данных.

## 6.2 Сообщения прикладных процессоров

Сообщения из прикладных процессоров, если речь идет о использовании PPDS баз данных до пользователя как правило не доходят. Смысл их в предупреждении об ошибках в формате входных данных, которые должны быть устранены перед передачей документа в основную базу данных. Однако, в этой части в систему заполнения базы данных и контроля качества вставлен не всегда надежный элемент - человек. Т.е. пользователь основной PPDS базы данных не гарантирован при работе с системой от общения со служебной информацией прикладных PPDL процессоров.

Первое, что нужно в таком случае, — отделить зерна от плевел, — опознать чье сообщение. По традиции оформления прикладных процессоров PPDS сообщения из них маркируются длинной строкой символов "$", например:

```
ARGUMENT IN TABLE 1     IS ABSENT
```

или более сложное сообщение:

```
EVOKE ERROR. PIPI(S-WAVE)0      NOT FOUND
REACTION SYNTAX ERROR NUMBER 2     IN MAGO
K- P --> PIPI(S-WAVE)0 SYG(1385P13)0
```

Оба приведенных примера — сообщения процессора контроля ввода данных базы данных REACTIONS.

Как видно из примеров, сообщения могут быть понятными, (как в первом случае), понятными наполовину, (как в описании ошибки в PPDL имени состояния), и, вовсе непонятными (описание процедуры, в которой ошибка задетектирована. В примере "MAGO"). Последние — отражение процесса постоянной отладки и усовершенствований, проводимых администраторами баз данных в доступной им части системы. В версии BDMS/4 стандарты на описания ошибок поступающие из процессоров не введены.

Кроме формата сообщение процессоров контроля данных отличается от системных тем, что, как правило, сопровождает поток входных данных, или описания запросов не встречаясь в управляющих инструкциях системе.

При использовании подсистемы GUIDE описания ошибок во входных данных становятся существенном элементом диалога при вводе в базу данных документа. Разобраться в источнике ошибки, в смысле описания здесь должны помочь встроенные в GUIDE самоописания, срабатывающие в любом контексте подсистемы.

# Глава 7

# Администрирование в рамках PPDS

Здесь мы рассмотрим поцесс сопровождения данных в рамках PPDS проекта при использовании СУБД BDMS. BDMS, как уже описано выше, позволяет конструировать структуру записи в базе данных, создавать базу данных, пополнять ее, редактировать размещенные в ней данные, представляет достаточно удобные возможности для доступа к данным.

На использовании этой СУБД строят свою деятельность несколько информационных центров географически достаточно широко разбросанных. Требования кооперации между ними наложило отпечаток на формирование системы, на саму процедуру работы с нею.

Выше рассматривались в основном случаи использования BDMS при сопровождении ОСНОВНЫХ баз данных PPDS. Но с помощью BDMS можно конструировать пользовательские базы данных, со структурой определенной пользователем под его задачу — "Микробазу".

В этой главе приводится краткое описание особенностей работы с ситемой. Некоторые практические результаты ее использования.

## 7.1 Базы и "микробазы" данных

PPDS база данных поддерживается администратором, специалистом задачами которого являются: определение границ предметной области отражаемой в база данных, выработка правил отображения предметной области в данных, т.е. язык описания данных, контроль за соблюдением кодировщиками-индексаторами правил кодирования, обеспечение полноты представления данных в базе, обеспечение ее использования.

Международный характер PPDS проекта дополнительно усложняет каждую из перечисленных обязанностей администратора (правда, кооперация умножает количество администраторов).

# Глава 8

# Дальнейшее развитие BDMS

По разным причинам, о которых уже говорилось выше, BDMS в течении какого-то времени будет оставаться действующей системой. Трудно оценить его возможное время жизни, но, приводимая ниже информация не есть отражение лихорадочных попыток его продлить. BDMS есть, работает, по сегодняшним требованиям к системам такого типа нуждается в усовершенствованиях, среди которых есть и такие, реализация которых сравнительно мало трудоемка. Просто в момент возникновения BDMS соответствующие концепции еще не существовали.

Ниже изложены предположительные дополнения в систему, (как правило об изменении, отмене каких-либо существующих возможностей или свойств речи нет). Порядок изложения — первыми следуют наиболее легкие по исполнению (с точки зрения авторов) изменения.

- REGENT

(REport GENerated Tools) — процедура создания выходных форм, документов. Идея внедрения в BDMS этой процедуры самая старая из идей его преобразования, до сих пор частично реализованная только на уровне пост-процессоров для отдельных типов выходов из баз данных, например CSBOOK, VOCBOOK, BILIST. Необходимо создать достаточно удобный язык описания формата вывода, по возможности нескольких уровней, в конечном варианте позволяющий учитывать возможность дальнейшей обработки выходных данных (например процессором текстов TeX).

- виртуальный терминал

Максимальное недружелюбие система проявляет на самом пороге, при попытке переключиться с одного типа терминала на другой. Отсутствие системы описания виртуального терминала в BDMS на сегодня сдерживает его дальнейшее развитие. Так что, возможно реализация этой системы будет одной из первых будущих новаций системы.

Введение виртуального терминала позволит усовершенствовать процедуру GUIDE, дополнить систему встроенным экранным редактором, ориентированным на структуру данных системы.

- защита данных

В BDMS'е не решена еще окончательно проблема защиты данных от случайной порчи при работе администратора по сопровождению данных. Однако просматривается довольно простой и надежный вариант ее решения, требующий незначительных затрат на рутинную программистику.

- спасение сета

Возможно и не самое главное для постоянно пополняемой базы данных, но не лищнее, и не лишенное смысла усовершенствование: возможность сохранить между работой отдельных заданий уже отобранные выборки. Хотя на их производство требуется сравнительно небольшое время, однако, можно представить ЭВМ настолько загруженной, что и этого времени будет жалко. По крайней мере для КОМПАС части коллаборации вопрос с экономией временм процессора не праздный.

- типы данных: DATE

Введение особого типа данных "ДАТА" не дань моде (современные СУБД по-видимому без подобных элементов данных госприемкой и не принимаются к рассмотрению). Достаточно с помощью инструкции INDEX ознакомится с теми ЭД в которых дата кодируется вручную, чтобы убедиться в насущности проблемы. (Например, дата кодирования 1338 год!)

- поиск по любому ЭД (даже неключевому.)

В принципе, нет каких-либо проблем в организации поиска по любому ЭД (не только по ключевому). Хотя по временным затратам такой поиск существенно обгоняет поиск по инвертированному списку значений, но при базах малого объема, для редко требуемых в запросах ЭД да и просто на всякий случае такая возможность интересна.

Естественно также расширить на все ЭД инструкцию INDEX.

- однозначный поиск

Старая мечта администраторов BDMS — организация в нем надежного однозначного поиска. В силу принятой в BDMS структуры данных возможны ситуации, при которых нельзя сформулировать запрос, приводящий к выбору только требуемых записей.

- сеть (базовый уровень дистрибутива = CERN)

Этот вопрос связан с последними по времени попытками использовать дистрибутив BDMS для работы по сетям. Принятая в BDMS/4 система описания файлового хозяйства вошла в противоречие с возможностями использовать его в сетях. Исправления тут требуются не очень существенные, однако довольно сильно может измениться существующая сегодня версия дистрибутива системы. Волей коллаборации это направление работы с BDMS сегодня интенсифицировано.

- внедрение в ядро системы некоторых функций процессора.
- расширение вспомогательных описаний
- библиотека программ (язык BDMS программирования) (OPY)
  - набор функций-примитивов системы.
- управление из данных:!!!:
- расширение FDT
- многобазовый BDMS (объединение баз данных: FDT - FDT BASE)

- динамическая память в BDMS. Бесконечный рекорд.
- реляционные вставки (+сетевые и иерархческие типы управления)
- селектирование микробазы из базы.
- расширение языка: редактируемая таблица запросов.
- расширение языка: экранная форма генерирования запроса для пользователя.
- процедура сравнения записей. Отбор совпадающих.
- процедура TEST:
  - объединение всех подсистем в подсистеме инициирования задания. (выбор через меню)
  - описание файлового хозяйства: по умолчанию, определяется системой.
  - контроль времени. Временные сообщения по ходу задания.
  - обработка прерываний.
  - подпроцессы. (для долгосчитающих и независимых процедур).
- использование VMS HELP и MAIL процедур из системы.
- ОПТИМИЗАЦИЯ (перевод части процессов на ассемблер. //диск, I/O//
- сортировка (русский алфавит)

# Глава 9

# Заключение

Прошедшие с момента создания BDMS годы отмечены бурным развитием программного обеспечения информационно-поисковых систем. Однако, BDMS, практически не меняясь, выжила, используется и независимо развивается несколькими лабораториями. Причины такой "живучести" в наличии созданных и отлаженных процессоров контроля данных физики высоких энергий, в больших объемах уже существующих BDMS баз данных, в международном сотрудничестве, требующем единообразия структур данных и СУБД (при ограничении на обмен современными технологиями), в достаточно высокой квалификации в области программирования "конечного пользователя" данной системы (что в более современных не является необходимым условием).

# Приложение A

# Документ

Примеры документов базы данных BDMS

```
          формат LIST
                 PRINT
                 DUMP
                 DOCUMENT
```

```
SC = AUBERT 74D;
R = PRL 33, 1404;
TY = JOUR;
D = 1974;
R. = COO-3069-271;
TY = PREP;
D = 1974;
AUTHORS;
A = J.J.Aubert;
A. = U.Becker;
A. = P.J.Biggs;
A. = J.Burger;
A. = M.Chen;
A. = G.Everhart;
A. = P.Goldhagen;
A. = J.Leong;
A. = T.McCorriston;
A. = T.G.Rhoades;
A. = M.Rohde;
A. = S.C.C.Ting;
A. = Sau Lan Wu;
I = MIT;
AUTHORS.;
A = Y.Y.Lee;
I = BNL;
```

```
T = Experimental observation of a heavy particle J;
AC = BNL;
DE = DAS;
REAC-DATA;
RE = P BE --> E- E+ X;
RE. = P BE --> J/PSI(3097) X;
PLAB = 19.000,20.000;
PART-PROP;
P = J/PSI(3097);
PP = EX;
PP. = MASS;
DEN = 76/06/12  10.31.21;
DCH = 78/06/11  09.48.23;
*E

AUBERT 74D

     PRL 33, 1404 (1974) (JOUR)
     COO-3069-271 (1974) (PREP)
     (SEE *PRL 25, 1523)
     (MORL TING 75B)

Experimental Observation of a Heavy Particle J

    MIT - J.J.Aubert, U.Becker, P.J.Biggs, J.Burger,
          M.Chen, G.Everhart, P.Goldhagen, J.Leong,
          T.McCorriston, T.G.Rhoades, M.Rohde, S.C.C.Ting,
          Sau Lan Wu
    BNL - Y.Y.Lee

    ACCELERATOR=BNL; DETECTOR=DAS; EXPERIMENT=BNL-XXXFV

P BE --> E- E+ X          19-20 GEV (PLAB)              MASS
P BE --> J/PSI(3097) X      ''                            ''

         J/PSI(3097)        EX,MASS

            ENCODED DEC 1974 BY FU
```

ПЛОТНЫЙ ДАМП

```
SC = AUBERT 74D;R = PRL 33, 1404;TY = JOUR;D = 1974;R. = COO
-3069-271;TY = PREP;D = 1974;AUTHORS;A = J.J.Aubert;A. = U.B
ecker;A. = P.J.Biggs;A. = J.Burger;A. = M.Chen;A. = G.Everna
```

```
rt;A. = P.Goldhagen;A. = J.Leong;A. = T.McCorriston;A. = T.G
.Rhoades;A. = M.Rohde;A. = S.C.C.Ting;A. = Sau Lan Wu;I = MI
T;AUTHORS.;A = Y.Y.Lee;I = BNL;T = Experimental Observation
of a Heavy Particle J;PR = BNL-XXXFV;RR = SEE *PRL 25, 1523;
RR. = MORL TING 75B;OR = PPF 7447;OR. = PPA 7502;SB = ENCODE
D DEC 1974 BY FU;AV = PPDC;AC = BNL;DE = DAS;REAC-DATA;RE =
P BE --$>$ E- E+ X;RE. = P BE --$>$ J/PSI(3097) X;B = 19-20 GEV
(PLAB);PLAB = 19.000,20.000;DD = MASS;PART-PROP;P = J/PSI(30
97);PP = EX;PP. = MASS;DEN = 76/06/12  10.31.21;DCH = 78/06/
11  09.48.23;*E
```

## РЕДКИЙ ДАМП

```
SC = AUBERT 74D;
R = PRL 33, 1404;
TY = JOUR;
D = 1974;
R. = COO-3069-271;
TY = PREP;
D = 1974;
AUTHORS;
A = J.J.Aubert;
A. = U.Becker;
A. = P.J.Biggs;
A. = J.Burger;
A. = M.Chen;
A. = G.Everhart;
A. = P.Goldhagen;
A. = J.Leong;
A. = T.McCorriston;
A. = T.G.Rhoades;
A. = M.Rohde;
A. = S.C.C.Ting;
A. = Sau Lan Wu;
I = MIT;
AUTHORS.;
A = Y.Y.Lee;
I = BNL;
T = Experimental observation of a heavy particle J;
AC = BNL;
DE = DAS;
REAC-DATA;
RE = P BE --$>$ E- E+ X;
```

```
RE. = P BE --$>$ J/PSI(3097) X;
PLAB = 19.000,20.000;
PART-PROP;
P = J/PSI(3097);
PP = EX;
PP. = MASS;
DEN = 76/06/12  10.31.21;
DCH = 78/06/11  09.48.23;
*E


                        PRINT
SC = AUBERT 74D;
R.1 = PRL 33, 1404;
  TY = JOUR;
  D = 1974;
R.2 = COO-3069-271;
  TY = PREP;
  D = 1974;
AUTHORS.1;
  A.1 = J.J.Aubert;
  A.2 = U.Becker;
  A.3 = P.J.Biggs;
  A.4 = J.Burger;
  A.5 = M.Chen;
  A.6 = G.Evernart;
  A.7 = P.Goldhagen;
  A.8 = J.Leong;
  A.9 = T.McCorriston;
  A.10 = T.G.Rhoades;
  A.11 = M.Rohde;
  A.12 = S.C.C.Ting;
  A.13 = Sau Lan Wu;
  I = MIT;
AUTHORS.2;
  A = Y.Y.Lee;
  I = BNL;
T = Experimental observation of a heavy particle J;
AC = BNL;
DE = DAS;
REAC-DATA;
  RE.1 = P BE --> E- E+ X;
  RE.2 = P BE --> J/PSI(3097) X;
  PLAB = 19.000,20.000;
```

```
PART-PROP;
  P = J/PSI(3097);
  PP.1 = EX;
  PP.2 = MASS;
DEN = 76/06/12  10.31.21;
DCH = 78/06/11  09.48.23;
```

# Приложение Б

# HELP

Вспомогательные описания системы и баз данных размещаются в текстовых файлах SYSTEM.HLP и <DBname.HLP> Данные в этих файлах разделяются на секции ключевыми словами.

Схема секции файла вспомогательных описаний.

| | |
|---|---|
| #NEWS | ← начало новой секции |
| Текст оповещения в свободном формате. | |
| & | ← выход из секции |
| #<TERM1> #<TERM2> ... | ← совпадающее начало разных секций |
| Описание в свободном формате | ← текст секции |
| #<TERMn> | ← начало секции n |
| & | ← флаг конца секции |

Ключевые слова или фиксированы, если они определяются системой, в зависимости от контекста, в котором требуется вспомогательное описание, или определены в тексте секции, как метки. Если контекстно зависимое ключевое слово отсутствует в соответствующем файле описаний, то система сообщает:

NO INFO ON: <ключевое слово>

Признаком ключевого слова является символ "#" в первой позиции строки.

Печать из файла на экран начинается по обнаружении системой необходимого ключевого слова, и, прекращается при появлении строки с меткой другого ключевого слова. Система сканирует весь файл, печатая на экран все секции, отмеченные указанным ключевым словом. Дальнейший поиск ключевого слова прекращается, если в первой позиции строки в секции стоит символ "&".

В текст описания можно ввести команду, переопределяющую ключевое слово:

%<символ><новое ключевое слово>

Символ "%" должен стоять в первой позиции строки. Если в пределах секции вводится сразу несколько возможных ключевых слов, система предлагает пользователю выбрать из них нужное.

Система при старте сканирует файлы самоописаний и автоматически печатает секцию, отмеченную ключевым словом "NEWS".

#<DBname> — ключевое слово заглавной секции описывающей соответствующую базу данных

#FDT<DE> — ключевое слово, описывающее соответствующий элемент данных в описании к базе данных. Делает эту секцию доступной при обращении из команды FDT.

Файл описаний можно разбить на несколько отдельных файлов. Начинающаяся с первой позиции строки инструкция

@{имя файла} [{имя секции}]

означает, что далее текст вспомогательного описания должен браться из указанного файла. Если имя секции во внешнем файле не указано, то остается текущее, если указано, то текущее временно заменяется указанным.

По концу "включенного" файла управление вновь возвращается к исходному. Символ "@" в первой позиции воспринимается как командный только в рабочей секции.

# Приложение B

# TEST

## B.1 Язык описания структуры документа PPDS

Основные PPDS базы данных можно считать единым информационным массивом, связанным не только общим языком, но и принципами формирования структуры документов, вплоть до практикуемых атрибутов элементов данных.

Во всех PPDS документах присутствуют ЭД: SC — краткий код документа, R — реферативная ссылка на источник, D, TY — подчиненные R ЭД описания даты публикации и ее типа, RE — PPDL описание реакции, BP — описание пучковой частицы, BT — имя частицы мишени, и другие.

## B.2 Кодирование данных (ОБЗОР)

В базе данных REACTIONS каждая зависимая переменная, например data point, характеризуется изучаемой реакцией, измеряемой наблюдаемой величиной, и значением некоторого числа независимых переменных.

В общем случае, независимая переменная Y, описывается для одной или более реакций и для каждой реакции для одной или более наблюдаемых и для каждой наблюдаемой как функции одной или более независимых переменных. Чтобы уточнить описание каждой точки данных, Y, необходимо задать имя и значение каждой независимой переменной, и специфицировать реакцию и наблюдаемую. В литературе данные в наиболее общей форме представляются в виде Таблиц (рисунки можно рассматривать как простейший тип таблиц). Заголовки колонок и ряд меток задаются независимыми переменными, включающими реакции и наблюдаемые, а внутренность таблицы содержит собственно точки данных. Язык кодирования точек данных в базе данных REACTIONS сконструирован максимально приближенным к виду, в котором данные описываются в литературе, т.е. в таблицах. Точки данных и независимые переменные, характеризующие их, кодируются в элементах данных подчиненных узловому ЭД TABLE в диаграмме структуры записи (см. Рис.1). Для наших целей здесь, мы ограничимся только ЭД TABLE, LOOP-OVER, N, V и Y.

Как сейчас будет ясно, для целей кодирования, удобно рассматривать реакцию и наблюдаемую величину как независимые переменные. При нормальном использовании, независимые переменные это такие величины как v, лабораторный импульс, энергия и др. Здесь они характеризуются именем, например THETA, P(RF=LAB), E(P=1,RF=LAB), и значением, например: 90, 100 МэВ, и т.д. Для упрощения процедуры кодирования мы добавляем к списку независимых переменных реакцию и имя наблюдаемой величины. При этом имена этих независимых переменных RE и YN соответственно, а их значениями являются цепочки символов, определяемые именем реакции и наблюдаемой.

Используя подобную концепцию независимой переменной мы предлагаем метод кодирования таблиц при рассмотрении различных примеров их заполнения. (В конце этого раздела приведен простейший пример.)

Рассмотрим таблицу следующей формы:

| | | | | | | | | | | | | | **TABLE 1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S | s1 | | | | | | | | | | | |
| | U | u1 | | | | | | u2 | | | | | |
| | W | w1 | | | w2 | | | w1 | | | w2 | | |
| X | Z | z1 | z2 | z3 | z1 | z2 | z3 | z1 | z2 | z3 | z1 | z2 | z3 |
| x1 | | y1 | y2 | y3 | y4 | y5 | . | . | . | . | . | . | y12 |
| x2 | | y13 | . | . | . | . | . | . | . | . | . | . | y24 |
| x3 | | . | . | . | . | . | . | . | . | . | . | . | . |
| x4 | | . | . | . | . | . | . | . | . | . | . | y47 | y48 |

Заглавными буквами обозначены независимые переменные, прописными (за исключением y) их значения. Эта таблица будет закодирована следующим образом:

```
TABLE;
LOOP-OVER;
N=Z;            V=z1; z2; z3;
LOOP-OVER;
N=W;            V=w1; w2;
LOOP-OVER;
N=U;            V=u1; u2;
LOOP-OVER;
N=S;            V=s1;
LOOP-OVER;
N=X;            V=x1; x2; x3; x4;
Y=y1; y2; y3; y4; y5; . . . y47; y48;
```

Хотя массив значений Y представлен как линейный вектор, рассматриваться как многомерный массив Y(Z,W,U,S,X). Порядок, в котором кодируются LOOP-OVER, N, V элементы значим, за исключением простейшего случая, когда V имеет одно значение. Таким образом, в приведенном выше примере, s1 ассоциировано с любым одиночным y1 и может кодироваться в любом порядке по отношению к порядку LOOP-OVER, N, V триплету. Напомним, что в BDMS чистые узловые элементы генерируются автоматически и при кодировании ЭД TABLE и LOOP-OVER не являются необходимыми.

Для переменной самого нижнего уровня, наиболее медленно меняющейся в приведенном примере: X, возможно чередовать x1 значения и y1 таким образом, что вводимые данные по форме напоминают кодируемую таблицу, (теперь необязательные ЭД TABLE и LOOP-OVER опущены):

```
N=Z;            V=z1; z2; z3;
N=W;            V=w1; w2;
N=U;            V=u1; u2;
N=S;            V=s1;
N=X;
V=x1;           Y=y1; y2; y3; y4; y5; ...; y12;
V=x2;           Y=y13; y14; ...; y24;
V=x3;           Y=y25; ...;
V=x4;           Y=...; y47; y48;
```

Имена и значения каждой независимой переменной от которой зависит зависимая, должны быть представлены в каждой таблице.

Пример TABLE.1 демонстрирует возможность описания реакции и наблюдаемых как для случая большого количества реакций, например, в статье сообщающей о получении значений поперечного сечения для многих различных реакций сразу, так и в случае большого количества различных наблюдаемых при только одной реакции, например коэффициентов полинома Лежандра. Или возможно кодирование одной наблюдаемой для одной реакции при различных импульсах и углах. Таким образом и название реакции и имя наблюдаемой и независимые числовые переменные в данном языке кодирования имеют одинаковый статус.

Рассмотрим таблицу, в которой несколько независимых переменных изменяются одновременно:

| | | | | | | | **TABLE 2** |
|---|---|---|---|---|---|---|---|
| | | | S | s1 | | | |
| | | | U | u1 | | | |
| | | | W | w1 | | w2 | |
| X1 | X2 | X3 | Z | z1 | z2 | z3 | z4 |
| x11 | x12 | x13 | | y1 | y2 | y3 | y4 |
| x21 | x22 | x23 | | y5 | . | . | y8 |
| x31 | x32 | x33 | | y9 | . | . | . |
| x41 | x42 | x43 | | y13 | . | . | y16 |

TABLE 2 будет закодирована:

```
TABLE;
LOOP-OVER;
N=Z;            V=z1; z2;
LOOP-OVER;
N=W;            V=w1; w2;
LOOP-OVER;
N=S;  U;        V=s1; u1;
LOOP-OVER;
N=X1;X2;X3;     V=x11;x12;x13;x21;x22;...;x43;
Y=y1; y2; y3; y4; ...; y16;
```

Или, что то же самое:

```
N=Z;            V=z1; z2;
N=W;            V=w1; w2;
N=S; U;         V=s1; u1;
N=X1; X2; X3;
V=x11; x12; x13;        Y= y1;  y2;  y3;  y4;
V=x21; x22; x23;        Y= y5;  y6;  y7;  y8;
V=x31; x32; x33;        Y= y9; y10; y11; y12;
V=x41; x42; x43;        Y=y13; y14; y15; y16;
```

В TABLE 2 три независимых переменных X1, X2, X3 меняются одновременно. Это кодируется с использованием трех генераций N под одной генерации LOOP-OVER. Если N многократно возникает таким образом, то V имеет группы значений такой же множественности как N и каждая группа ассоциируется с одной точкой данных таким же образом, как и единичная точка данных в примере TABLE 1.

```
1) N=A; V=a; N=B; V=b; N=C; V=c; N=D; V=d;
2) N=A; B; C; D;  V=a; b; c; d;
3) N=A; V=a; N.=B; V.=b; N.=C; V.=c; N.=D; V.=d;
```

Эти три метода дают одинаковый результат: т.е. единичное значение каждой переменной ассоциируется с каждой точкой данных в таблице. При этом, 1) узловой элемент LOOP-OVER будет генерироваться для каждой пары N и V, тогда как 2) и 3) будет создан только один ЭД LOOP-OVER. 2) и 3) полностью эквивалентные формы и дают более эффективный способ размещения данных в записи.

Ни на число независимых переменных, ни на число ЭД LOOP-OVER нет ограничений.

Если какой-то элемент таблицы не заполнен, то можно использовать знак "-" для заполнения этого элемента.

Специальный случай множественной генерации N под одним LOOP-OVER для одной наблюдаемой зависящей от более чем одной реакции, как для наблюдаемой SIG(реакция 1)/SIG(реакция 2). Кодируется он следующим образом:

```
N=RE(Q=R1); RE(Q=R2); V=REACTION1; REACTION2;
N=YN;           V=SIG(Q=R1)/SIG(Q=R2);
```

где используется квалифицирующий аргумент Q (см. X-NAMES).

### B.2.1 Пример таблицы

Дифференциальное поперечное сечение для упругого рассеяния при 50ГэВ/с табулируется по переданному импульсу.

```
TFP=T_2;
RE=PI+_ --$>$_PI+_P;
OBS=D(SIG)/D(T);
PLAB=50,50;
N=P_IN_GEV;              V=50;
N=RE;                    V=PI+_P_--$>$_PI+_P;
N=YN;                    V=D(SIG)/D(T)_IN_MUB/GEV**2;
N=-T_IN_GEV**2;
V = 0.075;               Y = 14812 +- 245;
V.= 0.085;               Y.= 13618 +- 233;
V.= 0.095;               Y.= 12580 +- 244;
     .
     .
     .
V.= 1.65;                Y.= 0.69 +- 0.43;
```

## B.3 Поиск по содержимому значения ключевого ЭД

В случае текстовых значений ключевых элементов данных возможность выбора только по полному или усеченному с конца значению КЭД может оказаться недостаточной или недостаточно удобной. Возможности выбора расширяются при использовании анализа содержимого значения КЭД. Выборка по содержимому производится, если при имени КЭД в стандартном простейшем запросе указывается квалификатор CONTAIN:

FIND <имя КЭД>/CONTAIN = условие выбора; ...

Простейшее описание условия выбора - контрольная цепочка символов совпадение которой с любой подцепочкой в значении КЭД является критерием выбора.

Символ "%" в контрольной цепочке отмечает положение "любого" символа. Например:

AGINENKO

AJINENKO

удовлетворяют условию выбора:

A%INENKO

Символом "*" отмечается положение "любой" подцепочки символов внутри контрольной цепочки символов. В понятие "любой" здесь включено: любая длина цепочки из любых символов. Например:

AZHINENKO AJINENKO AGINENKO

удовлетворяют условию выбора:

A*INENKO

Можно указать номера позиций символов в значении КЭД по которым, и только по ним должно осуществляться сравнение с условием выбора: зону контроля. С этой целью в условие выбора вводится следующее описание:

$: n - n - n - n_k ... n_{k+1} - n:$ [1]

где $n$ — номер позиции в цепочке символов КЭД.

Условие, при соблюдении которого конструкция [1] рассматривается как описание зон контроля.

$nnnnn_k n_{k+1}$ [2]

На этой схеме закрашены зоны контроля соответствующие указанным в [1]. При сравнении с условием выбора [2] представляется процедуре контроля как непрерывная цепочка символов:

$nnn.....n_k n_{k+1}$,

Если первый номер в [1] записан с минусом:

$: -n... :$

то им отмечается зона контроля от начала цепочки до позиции ???

Аналогично, если последний в конструкции номер не имеет "пары", то зоной контроля будет от $n_{k+1}$ до последнего символа значения КЭД.

В контрольной цепочке символов кроме "%" — "любого" можно ввести описание символа "одного из ..." При этом существенно используется порядок символов в ASCII кодировке.

Конструкция:

>A<Z

означает, что стоящий на ее месте в контрольной цепочке символ — одна из букв от A до Z, но не цифра и не какой-либо другой символ. Совмещая подобные конструкции можно указать допустимые "зоны" для возможных символов:

$> s < s > s < s....$ [3]

где $s$ — символ. Как и для описания [1] требуется выполняемость условия:

$sss...s$

Все описание [3] соответствует при этом только ОДНОМУ символу. "Зоны" для символов описываются по тем же правилам, что и "зоны контроля". Например:

<A

означает: "все не буквы".

В условие выбора можно указать цепочку символов, наличие которой в значении КЭД запрещает его выбор, условие "NOT". Формат такого описания:

#текст

Например, условие:

A#B

выполняется для тех значений, в цепочке символов которых есть хотя бы один символ "A" но нет ни одного символа "B".

В одном условии выбора можно с помощью логических операторов из простых условий сконструировать сложное. Логические операторы обозначаются символами: "$" — AND, и "&" — OR.

Пример такой конструкции:

ABC$DEF$1A

при которой критерием выбора является наличие в цепочке символов значения КЭД подцепочек "ABC" и "DEF" и "1A".

При описании условия выбора символы "*", "#", "$", "&" использованы в качестве команд языка описания. Если необходимо, чтобы эти символы участвовали в сравнении, их нужно печатать дважды. Например:

A**B>>C
описывает условие выбора
A*B>C
в котором уже ни "*" ни ">" не являются метками языка описания.

Кроме инструкции FIND процедура выбора значения КЭД по анализу его содержимого работает в инструкции INDEX/CONTAIN=...; (см. 5.1).

При формировании сложного запроса квалификатор /CONTAIN действует только на выбор по тому КЭД, при котором он указан:

FIND A/CON = AZH; AND I = SERP; **

В этом примере заказана выборка документов, содержащих ЭД "I" со значением "SERP", и ЭД "A" в значении которого встречается цепочка символов "AZH".

| | | Таблица меток |
|---|---|---|
| Команда | Функция | Примечание |
| % | любой символ | |
| | любая цепочка символов | ** "*" |
| $: n - n \ldots :$ | описание зоны контроля | |
| $< s > s \ldots$ | ограничивающее описание контролируемого символа | >> "$"<br><< "<" |
| #.....# | описание запрета выбора (NOT) | ## "#" |
| $ | пересечение условий (AND) | $$ "$" |
| & | объединение условий (OR) | && "&" |

ПРИМЕЧАНИЕ: процедура поиска по содержимому может быть использована и при выборе по значениям КЭД числовых типов. При этом данные о формате этих значений преобразованных для выбора в текстовое представление можно получить с помощью инструкции INDEX.

## B.4 Кодирование данных (ОБЗОР)

В базе данных REACTIONS каждая зависимая переменная, например data point, характеризуется изучаемой реакцией, измеряемой наблюдаемой величиной, и значением некоторого числа независимых переменных.

В общем случае, независимая переменная Y, описывается для одной или более реакций и для каждой реакции для одной или более наблюдаемых и для каждой наблюдаемой как функции одной или более независимых переменных. Чтобы уточнить описание каждой точки данных, Y, необходимо задать имя и значение каждой независимой переменной, и специфицировать реакцию и наблюдаемую. В литературе данные в наиболее общей форме представляются в виде Таблиц (рисунки можно рассматривать как простейший тип таблиц). Заголовки колонок и ряд меток задаются независимыми переменными, включающими реакции и наблюдаемые, а внутренность таблицы содержит собственно точки данных. Язык кодирования точек данных в базе данных REACTIONS сконструирован максимально приближенным к виду, в котором данные описываются в литературе, т.е. в таблицах. Точки данных и независимые переменные, характеризующие их, кодируются в элементах данных подчиненных узловому ЭД TABLE в диаграме структуры записи (см. Рис.1). Для нащих целей здесь, мы ограничимся только ЭД TABLE, LOOP-OVER, N, V и Y.

Как сейчас будет ясно, для целей кодирования, удобно рассматривать реакцию и наблюдаемую величину как независимые переменные. При нормальном использовании, независимые переменные это такие величины как v, лабораторный импульс, энергия и др. Здесь они характеризуются именем, например THETA, P(RF=LAB), E(P=1,RF=LAB), и значением, напрмер: 90, 100 МэВ, и т.д. Для упрощения процедуры кодирования мы добавляем к списку независимых переменных реакцию и имя наблюдаемой величины. При этом имена этих независимых переменных RE и YN соответственно, а их значениями являются цепочки символов, определяемые именем рекции и наблюдаемой.

Используя подобную концепцию независимой переменной мы предлагаем метод кодирования таблиц при рассмотрении различных примеров их заполнения. (В конце этого раздела приведен простейший пример.)

Рассмотрим таблицу следующей формы:

| | | | | | | | | | | | | TABLE 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | s1 | | | | | | | | | | | |
| U | u1 | | | | | | u2 | | | | | |
| W | w1 | | | w2 | | | w1 | | | w2 | | |
| X Z | z1 | z2 | z3 | z1 | z2 | z3 | z1 | z2 | z3 | z1 | z2 | z3 |
| x1 | y1 | y2 | y3 | y4 | y5 | . | . | . | . | . | . | y12 |
| x2 | y13 | . | . | . | . | . | . | . | . | . | . | y24 |
| x3 | . | . | . | . | . | . | . | . | . | . | . | . |
| x4 | . | . | . | . | . | . | . | . | . | . | y47 | y48 |

Заглавными буквами обозначены независимые переменные, прописными (за исключением y) их значения. Эта таблица будет закодирована следующим образом:

```
TABLE;
LOOP-OVER;
N=Z;            V=z1; z2; z3;
LOOP-OVER;
N=W;            V=w1; w2;
LOOP-OVER;
N=U;            V=u1; u2;
LOOP-OVER;
N=S;            V=s1;
LOOP-OVER;
N=X;            V=x1; x2; x3; x4;
Y=y1; y2; y3; y4; y5; . . . y47; y48;
```

Хотя массив значений Y представлен как линейный вектор, рассматриваться как многомерный массив Y(Z,W,U,S,X). Порядок, в котором кодируются LOOP-OVER, N, V элементы значим, за исключением простейшего случая, когда V имеет одно значение. Таким образом, в приведенном выше примере, s1 ассоциировано с любым одиночным y1 и может кодироваться в любом порядке по отношению к порядку LOOP-OVER, N, V триплету. Напомним, что в BDMS чистые узловые элементы генерируются автоматически и при кодировании ЭД TABLE и LOOP-OVER не являются необходимыми.

Для переменной самого нижнего уровня, наиболее медленно меняющейся в приведенном примере: X, возможно чередовать x1 значения и y1 таким образом, что вводимые данные по форме напоминают кодируемую таблицу, (теперь необязательные ЭД TABLE и LOOP-OVER опущены):

```
N=Z;            V=z1; z2; z3;
N=W;            V=w1; w2;
N=U;            V=u1; u2;
N=S;            V=s1;
N=X;
V=x1;           Y=y1; y2; y3; y4; y5; ...; y12;
V=x2;           Y=y13; y14; ...; y24;
V=x3;           Y=y25; ...;
V=x4;           Y=...; y47; y48;
```

Имена и значения каждой независимой переменной от которой зависит зависимая, должны быть представлены в каждой таблице.

Пример TABLE.1 демонстрирует возможность описания реакции и наблюдаемых как для случая большого количества реакций, например, в статье сообщающей о получении значений поперечного сечения для многих различных реакций сразу, так и в случае большого количества различных наблюдаемых при только одной реакции, например коэффициентов полинома Лежандра. Или возможно кодирование одной наблюдаемой для одной реакции при различных импульсах и углах. Таким образом и название реакции и имя наблюдаемой и независимые числовые переменные в данном языке кодирования имеют одинаковый статус.

Рассмотрим таблицу, в которой несколько независимых переменных изменяются одновременно:

| | | | | | | | TABLE 2 |
|---|---|---|---|---|---|---|---|
| | | | S | s1 | | | |
| | | | U | u1 | | | |
| | | | W | w1 | | w2 | |
| X1 | X2 | X3 | Z | z1 | z2 | z3 | z4 |
| x11 | x12 | x13 | | y1 | y2 | y3 | y4 |
| x21 | x22 | x23 | | y5 | . | . | y8 |
| x31 | x32 | x33 | | y9 | . | . | . |
| x41 | x42 | x43 | | y13 | . | . | y16 |

TABLE 2 будет закодирована:

```
TABLE;
LOOP-OVER;
N=Z;            V=z1; z2;
LOOP-OVER;
```

```
N=W;            V=w1; w2;
LOOP-OVER;
N=S;  U;        V=s1; u1;
LOOP-OVER;
N=X1;X2;X3;     V=x11;x12;x13;x21;x22;...;x43;
Y=y1; y2; y3; y4; ...; y16;
```

Или, что то же самое:

```
N=Z;            V=z1; z2;
N=W;            V=w1; w2;
N=S; U;         V=s1; u1;
N=X1; X2; X3;
V=x11; x12; x13;        Y= y1;  y2;  y3;  y4;
V=x21; x22; x23;        Y= y5;  y6;  y7;  y8;
V=x31; x32; x33;        Y= y9; y10; y11; y12;
V=x41; x42; x43;        Y=y13; y14; y15; y16;
```

В TABLE 2 три независимых переменных X1, X2, X3 меняются одновременно. Это кодируется с использованием трех генераций N под одной генерации LOOP-OVER. Если N многократно возникает таким образом, то V имеет группы значений такой же множественности как N и каждая группа ассоциируется с одной точкой данных таким же образом, как и единичная точка данных в примере TABLE 1.

```
1) N=A; V=a; N=B; V=b; N=C; V=c; N=D; V=d;
2) N=A; B; C; D;  V=a; b; c; d;
3) N=A; V=a; N.=B; V.=b; N.=C; V.=c; N.=D; V.=d;
```

Эти три метода дают одинаковый результат: т.е. единичное значение каждой переменной ассоциируется с каждой точкой данных в таблице. При этом, 1) узловой элемент LOOP-OVER будет генерироваться для кажтой пары N и V, тогда как 2) и 3) будет создан только один ЭД LOOP-OVER. 2) и 3) полностью эквивалентные формы и дают более эффективный способ размещения данных в записи.

Ни на число независимых переменных, ни на число ЭД LOOP-OVER нет ограничений.

Если какой-то элемент таблицы не заполнен, то можно использовать знак "-" для заполнения этого элемента.

Специальный случай множественной генерации N под одним LOOP-OVER для одной наблюдаемой зависящей от более чем одной реакции, как для наблюдаемой SIG(реакция 1)/SIG(реакция 2). Кодируется он следующим образом:

```
N=RE(Q=R1); RE(Q=R2); V=REACTION1; REACTION2;
N=YN;           V=SIG(Q=R1)/SIG(Q=R2);
```

где используется квалифицирующий аргумент Q (см. X-NAMES).

Пример таблицы:

Дифференциальное поперечное сечение для упругого рассеяния при 50 ГэВ/с табулируется по переданному импульсу.

```
TFP=T_2;
RE=PI+_ -->_PI+_P;
OBS=D(SIG)/D(T);
PLAB=50,50;
N=P_IN_GEV;               V=50;
N=RE;                     V=PI+_P_-->_PI+_P;
N=YN;                     V=D(SIG)/D(T)_IN_MUB/GEV**2;
N=-T_IN_GEV**2;
V = 0.075;                Y = 14812 +- 245;
V.= 0.085;                Y.= 13618 +- 233;
V.= 0.095;                Y.= 12580 +- 244;
     .
     .
     .
V.= 1.65;                 Y.= 0.69 +- 0.43;
```

REC-ID, значением которого является номер записи в базе данных. В структуре данных BDMS этому ЭД принадлежит роль связующего элемента между записью, как целое, и составляющими ее элементами данных. Являясь обязательной принадлежностью записи этот ЭД не представлен в документе. ЭД обладающие таким свойством называются ВИРТУАЛЬНЫМИ. REC-ID — единственный в системе СИСТЕМНЫЙ ЭЛЕМЕНТ ДАННЫХ. Все остальные задаются при конструировании базы данных.

ЗНАЧЕНИЕ ( = {текст} ;)

Этот атрибут ЭД не имеет имени. Его свойства определяются значениями других атрибутов элемента данных: TY, IPROC, KEY, и др. (см. ниже).

Элемент данных представлен в записи своим значением. В документе имя и значение ЭД объединяются выражением, общая форма которого:

{имя}={значение}; (7)

Значение элемента данных в документе — это цепочка символов всегда завершаемая символом ";". При необходимости вставить символ ";" в текст значения его нужно использовать дважды. Пробелы в начале цепочки символов значения и в конце, перед ограничивающим символом ";" во внутреннем представлении игнорируются. Если требуется передать начальные или конечные пробелы во внутреннее представление, то используется символ ПРИНУДИТЕЛЬНОГО ПРОБЕЛА "_".

ТИП значения элемента данных (TY = {text};)

Этот атрибут описывает системе правило перевода значения ЭД из внешнего текстового представления во внутреннее представление в записи. Во внутреннем представлении значение ЭД может быть:

| | | |
|---|---|---|
| а) | текстовой строкой; | TY = CHAR; |
| б) | целочисленным вектором; | TY = INTE; |
| в) | вектором действительных чисел; | TY = REAL; |
| г) | вектором чисел двойной точности; | TY = DOUB; |
| д) | чисто узловые, не имеющие значения; | TY = NODE; |

ЭД любого типа можно присвоить нулевое, пустое значение:

# Содержание