Britton Lee Technical Publications

BL700 OPERATION MANUAL

(8.0)

September 1987
Part Number 201-1078-008

This document supersedes all previous documents.

The information contained within this document is subject to change without notice. Britton Lee assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under license and may only be used or copied by the terms of such license.

Table of Contents

Preface

This manual is designed for use by the System Administrator or Operations personnel. It contains instructions on the use of the Britton Lee database server utilities CKDB and DFU.

System requirements are:

BL700 or database server

release 43 or higher IDM/RDBMS

The following documents should be available to the person running these utilities.

Referenced Documents

System Administrator's Manual (SAM)

This document is central in understanding the Britton Lee database server from a System Administrator's viewpoint. It describes backup and restore procedures, security, the system tables and troubleshooting techniques.

Maintenance Manual

The scope of this manual includes general preventive maintenance procedures, self–diagnostic features, and the removal and replacement of components. A troubleshooting section describing problem symptoms and solutions is also included.

Installation Manual

This document provides the information needed to load the utilities DFU and CKDB to the Britton Lee database server.

Checkdatabase

Introduction

Checkdatabase (CKDB) is a utility that looks for inconsistencies in the data on a Britton Lee database server. It is primarily a debugging tool, and it has useful applications both inside Britton Lee and at customer sites. Because it is intended for occasional use, it does not receive commands through the Britton Lee database server's channels; instead, it receives commands through the console port as does the DiskFormat Utility (DFU).

Verfication of Database

CKDB's primary function is to verify the physical structure of databases. Many sources of inconsistencies are examined, including malformed tuples, invalid indices, and incorrectly allocated data. You may check a single relation in one database or every relation in all databases on the Britton Lee database server or a range of items in between.
All problems are flagged with a severity code and a short description of each error is given. An error summary is printed after the scan of the database has been completed. Usually, inconsistencies are reported and corrective action is not taken. However, some types of errors may be corrected if you make an explicit request. Corrections are not performed if there is a possibility that CKDB may have misdiagnosed the problem.

General Purpose Diagnostic Tools

The other major CKDB functions aid in solving a problem once it has been detected. For example, CKDB can read blocks from the Britton Lee database server's disks. It can also perform simple, one-variable retrieves and read IDM tapes. These modes are intended to be used primarily by Britton Lee support and development personnel.

Miscellaneous Functions

CKDB's remaining capabilities are not closely associated, except that they are useful for specific debugging purposes. These modes allow you to do the following:

1. Verify the bad block remapping information on a disk.
2. Compare two databases for logical equivalence.
3. Check that databases on mirrored disks are identical or if they are not, which mirror is more consistent than the other.
4. List the databases on the Britton Lee database server.
5. List the relations in a database.
6. List the disks on-line.
7. Read blocks from an IDM tape.

Options

There are options that change how CKDB works. Some options affect many functions, while others affect only a single function. For instance, an option may allow a support engineer to obtain additional information about a relation that would not be useful to an end-user.

Using Checkdatabase

Introduction

CKDB communicates through the Britton Lee database server's console port, not through a channel. To establish communications through the console port, connect an RS-232 ASCII terminal directly to the console port using a null modem cable. Next, type in the appropriate commands and read the responses. You may be able to link a terminal of a host computer to the console port of the Britton Lee database server. An example is the PHI program idmboot which allows you to use an IBM PC as the console device.

Loading CKDB into the Database server

CKDB can either be loaded temporarily into Britton Lee database server memory or stored permanently on the system disk.

Storing CKDB in the System Database

CKDB is not always placed in the system database after the rest of the IDM/RDBMS code is loaded. Fileload will load CKDB only if there is enough free disk space. If there is insufficient space for CKDB, then perform the following steps:

1. Turn the front panel switch to RUN, which brings up the IDM/RDBMS code.

2. Extend the system database (the default demand is sufficient).

```
1) open system;
1) alter database system;
180 rows affected
1) sync;
1) exit;
```

3. Turn the switch to OFF, then to MAINT.

4. Run Fileload and load the file CKDB from the IDM/RDBMS system software tape or diskettes. For more information, refer to the appropriate sections on loading utilities in the *Installation Manual.*

Loading CKDB into Memory

If you cannot copy CKDB permanently into the system database, it is still possible to load it into memory directly from a software load device or the IDM tape. CKDB is very similar to DFU in this respect because you can run it directly from the distribution device via either:

```
<BL series> Filename: load kernal ckdb
```

or

```
<BL series> Filename: loadtape kernal ckdb
```

If you load CKDB using one of the methods above, then skip the following section which only applies to running CKDB off the disk.

Starting CKDB

1. If the Britton Lee database server is currently in RUN mode, turn the front panel key switch to SAFE and wait for the SAFE light to turn on so that all currently running commands may finish. If you want current running processes to be backed out, then turn the switch to MAINT, and wait for the MAINT light to turn on.

2. Turn the front panel keyswtich to OFF and then to MAINT.

3. After the slots messages appear, type:

   <BL series> Filename: kernal ckdb

   CKDB will respond with:

   ```
   Ckdb version <ver> <day> <month> <time>
   Type ? for options and functions
   Name of database:
   ```

Normal Operation

The first thing that CKDB asks for is the name of a database to check. If you enter the name of a database, CKDB examines every object in the database.

```
Name of database: system
Examining relation relation 1
    unique clustered index on relid
Examining relation attribute 2
    unique clustered index or relid attid
    .
    .
    .
system: examined 129 pages, 973 tuples, 42
relations, 21 indices
Name of database:
```

The name and relid of each relation will be printed. If a relation has any indices, the type and keys of each index will be printed as well. As CKDB moves to each new task, it reports the object it is checking. After CKDB is finished with a database, it displays the number of pages, tuples, relations, and indices that were examined. If any errors occurred, the numbers of errors will also be printed.

Usage Hints

Introduction

CKDB has several rules that help give it consistent user interaction. These rules apply most, if not all of the time; the exceptions are described in the appropriate sections below.

Command and Option Format

Commands in CKDB are prefaced with a "/" and are followed by a lower case letter, such as /q which runs a query. Options have a "-" followed by single letter; e.g., -v sets the Verbose option. "Normal words" are usually assumed to be the name of a database or a relation.

Blanks

Spaces may be used freely where they do not cause a word or number to be broken. Their only function is to delimit tokens; beyond that, extra "white space" is ignored.

Displayed Integers

CKDB prints numbers in decimal and octal and to help reduce confusion, it usually follows simple rules in printing integers. Integer values that CKDB displays are in octal if the number is preceded by the digit 0. All other numbers are printed out in base 10 (decimal).
The only exception to this rule occurs in page dump mode. Page dump mode prints the bytes of data from the page in octal without the leading zero; it is not possible to display 16 bytes on a single 80 character line if the 0 is included.
Except as noted, numbers requested by CKDB are assumed to be decimal. You can override this default by starting an octal number with a 0 or a hexadecimal with a "#".

Carriage Return

The <cr> key is interpreted by most sections of CKDB as a quit command. In general, entering an empty line causes CKDB to "pop" back one level. Typing

relation to examine: <cr>

causes CKDB to return to the top level prompt:

Name of database:

The following example shows how this feature can be used to correct errors on previous lines.

```
First database? wrongdbname /* wrong name*/
Second database: <cr>
First database? rightdbname
Second database? <cr> /* now can quit *
First database: <cr>
Name of database:
```

Entering <cr> does not return:

1. During a yes or no question. It will ask the question again.

```
Really reboot: <cr>
Please answer y or n
Really reboot?
```

2. When the prompt says, "<cr> to continue," as in dump and query modes. The prompt <cr> indicates that you would like to continue viewing more of the information that was just displayed.

3. At the top level prompt "Name of database:" the <cr> is ignored.

```
Name of database: <cr>
Name of database:
```

Unprintable ASCII Character

Occasionally, bytes that are outside the normal range of printable ASCII characters are displayed. This can happen while CKDB displays relation or attribute names or when a character attribute of a tuple is displayed. Data that has values in the range of octal 0 through 31 are displayed as ^<char>, where <char> is obtained by adding 0100 to the byte. Thus, a null byte is printed as ^@. The other unprintable ASCII character (delete, 0177) is displayed via ^?.
CKDB strips off the uppermost (0200) character bit before sending it to the console.

Interrupting a Command

When CKDB is running normally, it may be interrupted by pressing the <space> key on the console device. After a few seconds, CKDB stops whatever it is doing and requests the next command. Any characters typed before the next prompt is printed, are silently discarded. Wait until CKDB acknowledges the interrupt before entering the next command.

Current Database

CKDB has a "current" database concept. Generally, it is the database from which CKDB reads currently. There is a stack of current databases organized into a typical last-in, first-out order which is used with nesting commands. When a database is "popped" off the stack, CKDB identifies the current database for you.

```
Name of database: -r db

Relation to examine: /r otherdb
relation   1   attribute    2   indices    3   protect
4
query      5   crossref     6   transact   7   batch
8
host_user  9   disk_usage   10  blockalloc 11
users      13  descriptions 15
Current database now is db
Relation to examine:
```

Options

Introduction

CKDB has several options that allow you to customize the basic verification operation of a database. For instance, on a slow terminal the speed of CKDB might be limited by the time it takes to print the relation names and index fields. In this case, it can be undesirable to print the names. An option has been provided which prevents them from being displayed. There are several other boolean option flags that enable certain functions. Some of the information obtained by these functions is useful, mainly for remote diagnostics, although the information can be used by anyone.
Options may be enabled or disabled at any time except during a yes or no question. All combinations of options are allowed; precedence between conflicting options resolves conflicting requests.

List of Options

All options start with the character "-" and have a single character name. To obtain a list of the flags, their names, and current values, type -?

Option	Description	Default Value
-b	Begin a scan at a certain relid or relname	False
-c	Print pages as characters	False
-d	Enter dump page mode on error	False
-e	Echo terminal input	True
-f	Automatically fix correctable	False
-i	Examine a particular index	False
-l	/c option - compare logged relations only	False
-q	Quiet. Print only error messages	False
-r	Examine a specific relation	False
-t	Print pages as tuples	True
-v	Print much information about each relation	False

Setting Options

You can type a "-" and a list of single character option names, either by itself or before another command. Turn off an option by capitalizing the letter in the option list. To accomplish this, the terminal must be able to transmit in both upper and lower case.
Examples:
Examine some particular relations in the system database. Examine all databases with the "quiet" option on and the "single relation" option disabled.

Name of database: **-qR /a**

The same actions could be performed by entering:

Name of database: **-q**
Name of database: **-R**
Name of database: **/a**

Options are changed only by user commands; CKDB never modifies any options by itself. For instance, in the last example the "quiet" option is still in effect after CKDB has examined all the databases.

OPTION

```
Fix correctable errors -f False
```

There are many correctable errors that CKDB can safely correct. For instance, it can deallocate pages that belong to a relation that no longer exists. A complete list of correctable errors and how CKDB fixes them is provided in the section "Error Messages" in this manual.
There is one correctable error that CKDB does not print unless "fixing" is enabled. This error is the correction of the "tuple" and "page" counts in the "relation" relation. It is common for these values to be inaccurate because they are guaranteed correct only if the Britton Lee database server is always turned off and if you use the SAFE position of the front panel keyswitch properly. CKDB does not consider it an error for tuple and pages counts to be wrong, but it corrects them because it is convenient to do so. When one of these counts is updated, CKDB prints:

Changing tuple count from <old value> to <correct value>

or

Changing page count from <old value> to <to correct value>

NOTE

Do not use the -f option if you have experienced a system error (SYSERR) or suspect that your database is damaged in any way. This option may try to fix errors caused by a system failure which will further damage your data. We recommend that you run CKDB without the -f flag first to determine if it is safe to use the flag.

OPTION

```
Examine only certain indices -i False
```

When examining single relations, it is useful to be able to examine a particular index, especially if one is suspected of being in error. By typing:

Name of database: **-i 0**
Name of database: **system**

CKDB checks only the clustered indices (index id 0) of all the relations in the system database. Other indices are not examined, although all of the remaining checks are still performed. Note that this is a numeric option and must be followed by the end of the line.

OPTION

```
Compare only logged relations -l False
```

This option is used only by the compare database mode function. Refer to the /c command for a complete description.

OPTION

```
Quiet -q True
```

This option is used to print only error messages and user prompts. It is most useful when a slow device is used for console input and output.
It temporarily hides the "verbose" option -v. When the "quiet" option is turned off, then the state of the verbose option will again be apparent.

OPTION

```
Name specific relations to examine -r False
```

Sometimes only a few relations need to be verified. With this option enabled, CKDB asks for the name of the relations to be checked, one by one. As with the (-b) option, a relation may be specified either by its numeric relid or by its name. If a name is chosen, it is possible that more

Boolean Numeric Options

Most of CKDB's options are simply boolean; i.e., they may take on only two possible values (true and false). Several boolean options can be set at the same time, starting the list with a single "-", as shown above.
However, some options can have a numeric value as well as an implicit boolean one. If you follow the option name with a number, then the option is not only "true", but in addition, it has the numeric value specified. It is possible to set a numeric option to true but not give it an integer value. Turning a numeric option off removes any numeric value that it may have had.

Option Descriptions

The following option descriptions contain the meaning of the option, its one (1) character option name, and its initial value.

OPTION

```
Begin scanning at a certain relation -b False
```

When an entire database is checked, the relations are processed in ascending relid order. The "begin" option allows the scan to start somewhere in the middle of the database. After a database is named, CKDB examines this flag to see if it should ask for a starting point.

Name of database: **system**
Beginning relation: **blockalloc**

A relation may be specified in several ways:

1. The name of the relation. This may be ambiguous if there are two relations of the same name but with different owners.

2. The name, a colon (":") and the owner of the relation. This specifies the object uniquely.

3. The integer relid of the relation. This also identifies it uniquely. To leave this mode and return to the top level prompt, enter a blank line.

OPTION

```
Dump pages as characters -c False
```

The "character" option tells the dump page function of CKDB to display disk pages as printable ASCII characters. See the descriptions of dump mode (/d) for further details.

OPTION

```
Enter dump mode on error -d False
```

This option allows disk pages to be examined whenever an inconsistency is detected. After printing a short explanatory message, CKDB calls the dump page function (/d). Data related to the error can then be examined to obtain the complete description of the problem.
Upon returning from dump mode, CKDB continues from the point at which the error occurred.

OPTION

```
Echo terminal input -e True
```

If a terminal is connected directly to the Britton Lee database server console port, you should probably enable the "echo" flag. If the console port is connected to a host computer that does its own terminal echo, it may be desirable to turn off the Britton Lee database server echo.

than one relation will be verified because two users may have relations with the same name. Any number of relations may be examined one at a time. Entering an empty line returns CKDB back to the top level. The allocation of each relation will be examined after the data and indices are verified.
If the database has more than 1000 relations in it, the system catalogues relation, attribute, and indices then examines them automatically. CKDB avoids using the indices on these relations unless it knows that they are good. However, it uses them often enough that the examination would be slowed significantly if it could not take advantage of their indices.

OPTION
`Print pages as tuples -t True`

The "tuple" option indicates that CKDB should display disk pages as tuples. See the dump mode (/d) description for further details.

OPTION
`Verbose -v False`

Print the name and relid of each relation and the numerical ID and status of each index. Enabling this flag causes more information from the relation and indices relations to be printed.

```
Relation to examine: -v relation
Examining relation relation 1
    first <01:014:100> last <01:014:100> type S
    status 0327 tuples 54 pages 7


2 empty data pages
unique clustered index on relid
    status 0203 root <01:014:103> itulten 5 card 1
    No root


unique non-clustered index 1 on name owner
    status 01 index 1 root <01:014:103> ituplen 21 card 1

Relation to examine: -v
Relation to examine: 1
Examining relation relation 1
unique clustered index on relid
unique non-clustered index 1 on name owner
Relation to examine:
```

Functions

Introduction

There are several modes of CKDB which do not directly check the physical structure of a database, but nonetheless are useful features. All of these sub-functions are of the form /<letter>; some of the commands optionally accept a database name after the function name. These functions may be called upon at almost any point where CKDB accepts input so that a call to one can be nested inside a call to another. Nested calls to the same function are NOT allowed.

Quick Reference Table of Functions

/a	All Databases
/c	Compare Database
/d	Dump Page Mode
/l	List Databases
/m	Checks Halves of a Mirror
/o	Print Disks On-line
/q	Execute Queries
/r	List relations
/s	Scan Disks
/t	Tape Mode
/x	Exit from CKDB

All Databases /a

Every database on a Britton Lee database server may be verified by typing:

Name of database: **/a**

CKDB will print the names of every database before it is examined. This is only valid from the top level prompt:

Name of database:

Compare Database /c

For the verification of channel drivers and the dump, load and rollforward commands are useful to know if two databases are logically identical. Compare database mode is entered by entering /c instead of a database name at the top level prompt:

Name of database: **/c**

CKDB responds with

First database ? **db**

and then

Second database ? **backup**

CKDB will then compare each relation or file in the first database with its counterpart in the second database, on a tuple-by-tuple or byte-by-byte basis.

```
relation    1
attribute   2
    .
    .
```

It does not try to find the minimal subset of differences which would make the objects equal, so a single tuple inserted into the beginning of one relation is all that is needed for CKDB to think that they are vastly different.
The -r and -b options may be used to compare only certain relations in the databases.

Comparing relations

Tuples from each relation are fetched and compared one at a time. The tids and attributes of tuples which differ are printed.
CKDB stops checking a relation when it finds five mismatching tuples; it then prints

Too many differences

and continues to the next relation.

Comparing files

The comparison of files stops as soon as a single byte difference has been detected, as in:

<filename> differs at <offset> <byte1> <byte2>

Relations Which May Be Compared

A relation or file in one database can be compared to its equivalent in another database only if relations have:

- the same name
- the same owner
- the same type of attributes
- the same number of attributes

The relations do not need to have the same relid.
If a relation exists only in the first database, CKDB will issue the message

<name> not common to both databases

and continue to the next relation in the first database. If the relation does exist in both databases but the attributes are not identical:

<name> has different attributes in the second database

Relations which are only in the second database are silently ignored.

Relations Which Cannot Be Compared

There are some objects which, even though they meet the above restrictions, cannot usefully be compared.

System Catalogues

Some system relations, such as blockalloc and disk–usage, are not comparable. These relations depend upon the physical layout of the blocks on the disk and are guaranteed to be different across databases. CKDB also ignores the disk–dependent fields of the relation and indices relations ("firstpg," "lastpg," and "rootpg") so they may be compared meaningfully.

Special relations

The text of views, stored programs, and stored commands is stored in the system query relation. Equivalence of these objects is checked when the query relation is compared. Compare mode silently skips over these objects.

Options For Compare Mode

Compare Only Logged relations -l False

The logged-only option makes CKDB skip relations which are not logged thus avoiding spurious error messages. If one database is the output of a rollforward, and another is the original database, non-logged relations that were changed will not be identical. When a non-logged relation is encountered, the message:

<relation> is not logged

will be printed, and the comparison will continue to the next relation.

Enter Dump Mode

Turning on the -d options indicates that CKDB should enter dump pages mode (/d) when a pair of tuples do not match. After returning from dump mode, the comparison will continue.
The "begin" option -b can be used to start the comparison at a certain relation. This is useful in skipping over relations which have previously been found to match. Similarly the "single-relation" can be used to compare only specific relations.

Dump Page Mode /d

This function allows a disk page to be looked at in a variety of formats. It prompts with:

Page number to read:

Now you need to know how to tell CKDB which page to read.

Specifying a Page

Every block on a Britton Lee database server is identified by a pages number which can range from 0 to 2^24-1. It can be specified in the following ways.

1. It can be a three byte page number with each byte being given separately. Much of IDM/RDBMS uses these three byte page numbers to identify a disk block.

   Page number to read: **0 51 227**

   By default these numbers are interpreted in octal; you can specify hexadecimal by preceding each number with a "#".

   Page number to read: **0 #21 #97**

   There is no way for them to be entered as three separate decimal numbers.

2. A single long page number. This one is decimal by default since many Britton Lee database server error messages (hard disk errors and syserrs) print out block numbers as a single long decimal integer. All of the following are equivalent to the previous example:

Page number to read: **020627**
Page number to read: **8599**
Page number to read: **#2197**

3. A four byte tuple identifier, or tid. The first three bytes of a tid is a page number; the last byte uniquely identifies the tuple on that page.

4. A physical block specification consisting of the following things.

   - disk controller number (ranges from 0 to 3)
   - disk drive number (ranges from 0 to 15)
   - cylinder of the block
   - track of the block
   - sector of the block

   EXAMPLE:

   Page number to read **c 0 5 47 6 18**

   The numbers listed above default to base 10.

   This type of page identifier is printed by the low level disk error routines.

   Disk hard err code xEOOO
   cmd 3, ctlr 0, drive 1, cyl 35, head 7, sec 1
   errcode 100, errstat 97, phyaddr 163800
   cmd was retried and failed.

5. A "b" in front of any of the previous ways to a page number. Occasionally, the information that you are interested in is not on the page itself but in its blockalloc tuple.

   <cr> to continue: **b 8599**

   If you accidentally entered dump mode, just type a <cr> to return back to the previous mode.

Checking Out Page Before Reading It

CKDB finds out several interesting things about a page just before reading it. If the page does not belong to the current database it will say so.

Page number to read: **0 23 303**
<0:23:303> belong to testdb (dbid 8)

When the page to read is on a mirrored drive, CKDB will ask you from which drive it should read.

Page number to read: **0 23 303**
<0:23:303:> is mirrored on ctlr 0 drive 0 and ctlr 1 drive 0
Use first drive? **n**
Using second drive

It reads the physical sector header of the page to see if it has been remapped to another page.
If the page number is on the same as the page that was specified, it prints:

Ask for <3:124:340>, got <3:127:103>

It looks at the "mode" field of the page's blockalloc tuple for one of the following cases:

02	the page is free
020	the page was marked bad by DFU and presumably

	contains a media defect
0104	a file data page
0220	a bad page which is remapped to another block
0204	a good page to which a bad block is remapped

If none of the allocation modes is found and the mode does not have the used (04) bit on, then the page is allocated incorrectly.

Display Format

The data from a page may be printed in a variety of formats. The first 14 bytes will always be printed as the internal page header (pageid, status, nextpg, nexttno, relid, freeoff, and level). The rest of the page may be displayed as octal, bytes, characters*, or tuples. By default they are displayed as 12 rows of 16 octal bytes. There are two options used only by dump mode to determine how the data of a page should be printed.

Print as Characters -c False

If the -c option is set then bytes that are printable ASCII or EBCDICcharacters will be shown as characters. Data that is outside the printable range of <space> to "~" will be displayed in octal. Bytes that contain values between 0 and 7 will be preceded by a "\" to distinguish them from the digits "0" to "7" (values 060 - 067).

Print Tuples -t True

The -t tells CKDB to try to print the page as a sequence of tuples. The tuple format is similar to that of the IDL or SQL parser. All the attributes of the relation are printed first, followed by a few tuples worth of data. To print tuples, CKDB must be able to get the attributes of the current page, which is quick and easy to do for data pages. However, it is not as efficient to find the attributes of a particular index page; CKDB may have to search all the indices of the relation in order to find out to which index a particular page belongs. It estimates how may index pages there may be; if it thinks that it will take a long time to search them all, it will ask whether you want to wait for it to find the index.

```
There are 23 indices to search
Continue?
```

This is the "indid" attribute of the indices relation. The indid of a clustered index is always 0; non-clustered indices are numbered from 1 to 250. When a non-clustered index is examined, its indid is printed before the keys of the index, as in:

```
Examining relation relation 1
    clustered index on relid
    non-clustered index 1 on name owner
```

If for any reason CKDB is unable to find the attributes, the tuple area of the page will be displayed as characters or octal bytes. Attributes may be unavailable because the database has been destroyed, the database is locked for a load or rollforward, or the relation has been destroyed.
If you elect to continue searching the indices, CKDB will print out the indid and attributes of the index when it is found. If you decide against waiting, each page is displayed as if the -t option were not turned on. However, all is not lost. If you know to which index this page belongs, you can tell CKDB by specifying a decimal index identifier after the -t, as in:

```
<cr> to continue: -t 3
```

Whether or not the attributes were found, the tuple number table at the end of a data page will be displayed as decimal offsets to tuples from the start of the page.

* Characters are printed in the mode appropriate to the database from which the page was read. Thus, EBCDIC characters are translated to their ASCII equivalents before they are printed.
The "tuple" option takes precedence over the "character" option.

Looking At a Page

Disks blocks on a Britton Lee database server contain 2048 bytes. It is not practical to print the whole page out at a time. Dump mode tries to print out about half a screenful of data. After printing those twelve lines*, CKDB will respond with:

<cr> to continue:

Type ? here to get the list of dump mode sub-commands. These commands are local to dump and are only recognized after you have read the page.

j<n>	jump to offset <n> in the page
f	follow the next page pointer in the page header
l	list next 10 page headers
p	find previous page
q	run a query on current pages relation
t<n>	go to tuple number <n> on this page
x	exit from this page
.[+ -]<n>	go forward or backward <n> pages
<pno>	exit from this page, read another

A command that causes another page to be read (f,p, or <pno> may be followed by another command, such as:

<cr> to continue: **fj 1000**

to read the following page and then start displaying from the middle of it.

Jump to Offset - j

The "jump" command allows data on the page to be seen in any order. The decimal number after the j (and optionally any blanks) should fall in the range 0 to 2047. If the offset given points into the middle of a tuple, CKDB will adjust the offset so the whole tuple will be printed. Otherwise, CKDB will round the address down to a 16 byte boundary; i.e. jumping to 37 will cause CKDB display from offset 32.

Follow Next Page Pointer - f

A page header contains a "next page" field. This page is logically related to the current page, and it is common to follow a chain of these pointers down a linked list of pages. The end of the list is marked by a next page of "<0:0:0>". CKDB prints:

<cr> to continue: f

On the last page it prints:

<cr> to continue:

if it is told to go beyond the end.
The f may be followed by a count of the number of pages to read; by default it goes forward one page.

* Only six lines are displayed if the "quiet" option is set.

List Next Ten Page Headers - l

Just as it is useful to follow the next page pointer, it is sometimes desirable to look at a list of page headers.

```
<cr> to continue: 1
<0:3:5> nextno 012 freeoff 48 stat 021
<0:3:6> nextno 013 freeoff 66 stat 021
<cr> to continue:
```

CKDB prints up to ten page headers. If the relid or level of the next page does not match the previous page, CKDB will note that and stop. It also indicates where there are more pages after the tenth, as in:

```
<cr> to continue: 1
<0:3:17> nextno 012 freeoff 48 stat 021
<0:3:20> nextno 012 freeoff 50 stat 021
                .
                .
                .
<0:3:35> nextno 012 freeoff 46 stat 021
<0:3:36> nextno 013 freeoff 66 stat
021...more
<cr> to continue:
```

indicating that there are more pages following.

Find Previous Page - p

Disk pages in the Britton Lee database server are grouped together in lists linked singly. Occasionally, it is desirable to go backwards in that list to find the page that points to the current one. Since there is no backwards pointer, there is no quick way to go back. The p command tells CKDB to search the entire database for a page which are allocated to the same relation. If it does not find a "previous page" at this point, CKDB will ask:

```
Try unallocated pages?
```

Enter y or yes and CKDB will look at all the other pages in the database. CKDB must use blockalloc to restrict its scan of the disk. If the database was not opened, CKDB prints:

```
Database not opened.
```

and stays at the current page. It does not try to find the previous page.

Queries - q

Dump mode allows you to run queries on the relation to which the current page belongs. You may restrict the queries to a subset of the relation, i.e., starting at the current page and going till the end of the data pages list. Refer to the /q mode description for further details.

Go To Specific Tuple - t

You can jump to a specific tuple on a data page with the t<n> command. Here <n> is the (octal) tuple number of the tuple that you want to look at.

```
<cr> to continue: t 27

          tno  mode       check stat      relid
152       027                0
158       030                0
```

```
164        031              0
165        032              0
166        033              0

<cr> to continue:
```

Go To Current Page +/- Some Amount

This command takes the page number of the current page, adds or subtracts the number given after the ".", and goes to that page. It is not a duplicate of the f and p commands. They read along a linked list of pages; the command does not care whether the next higher block number is the next page of the current block. If you are currently on page 3150,

```
<cr> to continue: .-20
```

is equivalent to:

```
<cr> to continue: 3430
```

If the current page is on a mirrored disk, the next page will be read automatically from the same drive. CKDB will not ask you whether or not it should read from the first drive of the pair or the second drive.

Exit From Page - x

To stop looking at this page, type:

```
<cr> to continue: x
Page number to read:
```

CKDB will return to the top level of dump page mode and ask for the next page to look at. Once at the prompt:

```
Page number to read:
```

entering <cr> will return out of dump mode.
Another way to leave is to go the end of the page and enter an empty line. The prompt changes here to inform you that you are about to "fall out" of dump mode.

```
2031 514 508 502 496 490 478 472
<cr> to leave page:
Page number to read:
```

At this point, either x or <cr> returns you to first dump mode prompt.

List databases /l

Print a list of all the databases that are on this Britton Lee system.

```
Name of database: /l
Databases on this IDM:

att foo    jwinery          mfgdb          mike
mike2          sprtest      system  vino2  workdb
Name of database:
```

Check Halves of Mirror /m

This command can be used on mirrored databases which "mirror compare" of DFU has found to be inconsistent. It can temporarily un-mirror and re-mirror a drive so that CKDB can subsequently be run

on all possible combinations of disks. You can then select the best set of drives and "mirror copy" from them.
Finding the best set of drives is somewhat complicated. The steps involved are:

1. Determine which databases are on the suspected drives. If you do not remember which they are, you may query disk_usage of each database with the /l function.

2. Find out the controller and drive numbers of the accessible disks with the /o command.

```
Name of database: /o
Ctlr 0 drive 2 disk160a 45091-118350   814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 2 disk160a 45091-118350   814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 0 disk160b 118351-191610  814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 0 disk 160b 118351-191610 814(823)M cyls 10 heads 9 sectors
Name of database:
```

3. Select the drive(s) that you wish to check first; it does not matter which disk you choose. In this example we shall turn off the mirroring on the second disks of each pair, so that only the first one will be used.

```
Name of database: /m 1 2
Ctlr 0 drive 2 disk160a 45091-118350   814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 2 disk160a 45091-118350   814(823)  cyls 10 heads 9 sectors
Ctlr 0 drive 0 disk160b 118351-191610  814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 0 disk 160b 118351-191610 814(823)M cyls 10 heads 9 sectors
Name of database:
```

Notice that the "M" is not present in the second drive any more; this means that the disk is not currently mirrored.

4. Toggle the mirroring on the drives so that we get the desired set of disks. In this case:

```
Name of database: /m 00
Ctlr 0 drive 2 disk160a 45091-118350   814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 2 disk160a 45091-118350   814(823) cyls 10 heads 9 sectors
Ctlr 1 drive 0 disk160b 118351-191610  814(823)M cyls 10 heads 9 sectors
Ctlr 0 drive 0 disk 160b 118351-191610 814(823) cyls 10 heads 9 sectors
Name of database:
```

Now CKDB will only see the data that is on the disks at ctlr 0/drive 2 and ctlr 1/drive 0. Notice that we turned off the mirroring of the third drive. Because the IDM/RDBMS wants the drive that thinks it is mirrored to be first, CKDB exchanged the order of the third and fourth drives.

5. Check the databases of specific relations which reside on the blocks numbered 45091 to 191610. The "fix" option -f should be off, because CKDB may report some spurious errors if you did not happen to chose the "correct" set of disks. The -f option will be turned off if it is on and the /m command is used.

   You must run mirror copy mode of DFU after using this command and checking any relations or databases.

Print Disks On-line /o

To find out which disk drives are connected to this Britton Lee database server use the /o command. It prints a list of both accessible and inaccessible disks with their drive characteristics and whether or not they are mirrored. A "foreign" disk that actually has a disk name may be an out-of-date mirror, or it simply might need to be merged into the system with DFU.

```
Name of database: /o
Ctlr 0 drive 2 disk160a 45091-118350  814(823)M cyls 10 heads 9 sectors
Foreign disks
Ctlr 0 drive 0 disk 80a 1-36450  810(823)M cyls 5 heads 9 sectors
Name of database:
```

The fields are:

1. the disk controller number (Controller 0 is the one in the lowest slot number, controller 1 is the one in the next lowest, etc.)
2. the disk drive number, which ranges from 0 to 15
3. the name of the disk
4. the lowest block number on the drive
5. the highest block number on the drive
6. the number of cylinders available for use by the database management code. The actual number of cylinders are on disks formatted without a remap zone (release 25 DFU or earlier).
7. The actual number of physical cylinders of the drive. Only remap zone disks have this field. If it is followed by an "M", the drive thinks that it is mirrored.

Execute Queries /q

It would be a tedious job to find a particular tuple in a relation just by using dump mode, looking at tuples one at a time until an interesting one is found. Fortunately, CKDB can execute a subset of one variable retrieves. This is a sub-mode which, like "dump" mode, can be entered at any time.

Entering Query Mode

Entering query modes involves selecting the database to use; once inside, many different relations may be retrieved without leaving this mode.
There are two ways to start querying:

1. On a regular command line where options and functions are recognized, type:

   ```
   Name of database: /q[<database>]
   Relation to query:
   ```

   to enter query mode. The database name is optional; by default the current database is chosen. CKDB will ask for a database name if there is no current database.

2. While looking at a page in dump mode, the command

   <cr> to continue: **q**

   may be used to try to run a query on the relation to which the current page belongs. If it is a data page, CKDB will ask whether the query should start or end on this page.

   <cr> to continue: **q**
   Start on this page? **y**
   End on this page? **n**
   1) attname relop value:

   Respond "no" to use the first (or last) page of the relation; enter "yes" to start (or end) on the current page; type "<cr>" to either question to return out of query mode.

   In either case, the attributes of the relation must be obtainable. Both dump and query mode have the same restrictions about attributes; refer to the discussion about the "tuple" option, -t for further information.

Selecting the Query Variable

Query mode only allows single variable queries. The relation to use is specified when query mode is entered:

Name of database: **/q testdb**
Relation to query: relation
1) attname relop value:

You can select another relation by entering an empty line at this point:

Name of database: **/q testdb**
Relation to query: relation
1) attname relop value: <cr>
Relation to query: **attribute**

Qualifications

CKDB can run queries of the form:

```
retrieve (x.att 1, x.att 2,...)
    where (x.a <relop> const 1) and
                .
                .
                .
                (x/b <relop> const 2) and
                (x.c <relop> const 3) or
                (x.c <relop> const 4) or
                (x.d <relop> const 5) go
```

The general format allows a string of "and" clauses followed by a string of "or" clauses; either or both may be empty. All of the "and" clauses must qualify; at least one of the "or" clauses have to be satisfied.
Once the relation has been selected, CKDB asks:

1) attname relop value:

This is the prompt for query mode. Either simple clauses (restrictions) or commands which modify the target list (projections) may be entered here. The order of any restrictions is immaterial. Type <cr> or go to execute the query.

AND Clauses

The attribute names (a, b, c, or d in the example above) should be one of the attributes of the relation.
The relational operator may be one of the following:

= or ==	attribute equality
! or !=	attribute inequality
>	attribute > constant
>=	attribute >= constant
<	attribute < constant
<=	attribute <= constant

Table 1-1: Relational Operators

It must be separated from the surrounding tokens by at least one blank. How the constant is entered depends upon the type of the attribute. One byte, two byte, and four byte integers are assumed (by default) to be decimal. Character strings start at the first non-blank after the relop and continue until the length of the attribute has been exceeded, the end of the line has been reached, or a blank has been seen. A character string that contains embedded blanks may be entered as a double-quote delimited string. Binary, bcd, and floating point data must be entered as a list of octal numbers, each one being the next byte in the binary string. For example:

attname relop value: **relid >= 21**
attname relop value: **type = T**
attname relop value: **firstpg > 0 16 262**

searches for all transaction logs which start after page "0 : 16 : 262>".

OR Clauses

These are differentiated from "and" clauses by the first character of the line. If it is a "|" then the rest of the line is added to a list of or clauses, otherwise it is treated as an and clause. The "|" must be separated from the attribute name by a blank or tab. The following query retrieves the stored commands or stored programs owned by Britton Lee database server user 23.

attname relop value | **type = P**
attname relop value | **owner 23**
attname relop value | **type = C**

Changing the Target List

Normally, all the attributes of the relation will be displayed. This can get a bit tiring especially on a slow console port. By typing:

1) attname relop value: **only <att1> <att2>...**

The domains mentioned will be the only ones displayed. Enter:

```
1) attname relop value: only
using all attributes again
```

to retrieve all the attributes again.
Attributes can also be excluded from the target list by typing:

```
attname relop value: not <att1> <att2>...
```

prevents the specified domains from being printed. Attributes which have been removed from the target list may be used in the qualification.

Count

The count command removes all the attributes from the target list. No tuples will be printed; however, it will display the number of qualifying tuples.

Correcting a Query

If a mistake has been made in the query, it can be entered again by typing:

```
3) attname relop value: reset
1) attname relop value:
```

and re-typing the query.

Queries from Dump Mode

Retrieves initiated from dump mode can be limited to a subset of the relation with the q command.

```
<cr> to continue: q
Start on this page: y
End on this page? n
```

```
<cr> to continue: q
Start on this page? y
End on this page? n
```

Starting and ending page numbers or tuple identifiers may be specified before the qualifications are given. If you give a page number, the scan starts at the first tuple on the page (or ends at the last tuple on the page). CKDB checks that the tids point to data tuples of the current relation before using them.

```
<cr> to continue: q
Start on this page? 0 244 53
End on this page? y
```

If the "q" was typed accidentally, just enter "<cr>" to return to dump mode.

```
<cr> to continue: q
Start on this page? <cr>
<cr> to continue:
```

List relations /r

Prints the names and relids of all the relations in a database. By default, CKDB uses the current database; another may be specified explicitly:

```
Relation to examine: /r otherdb
relation    1    attribute    2    indices    3    protect
4
query       5    crossref     6    transact   7    batch
8
host_user   9    disk_usage   10   blockalloc 11
users       1    descriptions 15
Current database now is system
```

Initially "system" is the current database.

Scan Disks

This function reads each zone of every accessible disk, looking for fragments which are allocated to non-existent databases, as well as fragments which are disjoint from their databases.

```
Name of database: /s
Ctlr 0  drive 2  disk160a 45091-118350 814(823)M cyls 10 heads 9 sectors
407 zones
Zone 0
Zone 10
.
.
Zone 400
Ctlr 0 drive disk160a 45091-118350  814(823)M cyls 10 heads 9 sectors
Zone 0
.
.
Zone 400
Name of database:
```

Scan disks uses disk_usage to determine whether a fragment is connected; if it is suspect, CKDB should be run on all the databases via:

```
Name of database: /a
```

before using /s to look for missing regions.
Scanning a disk also checks all the bad block remapping information. The physical sector header of a bad block contains the alternate cylinder, head, and sector of the block that is used to store the data of the page. There are many consistency checks done to verify that nothing is wrong with the remapping information.
To scan only a single disk enter the drive name:

```
Name of database: /s
Ctlr 0 drive 2 disk160b 45091-118350 814(823) cyls 10 heads 9 sectors
407 zones
Zone 0
Zone 10
.
.
Zone 400
Name of database:
```

Tape Mode /t

You can examine IDM tapes with CKDB's tape dump mode /t. It is similar to page mode except that it reads a tape drive instead of a disk. Its commands are:

```
Tape mode: ?
b move backwards a block
d display current block
f forward to a file mark
h search on page header
l list tape buffers
n go next block
r rewind tape
s seek to a block #
Tape mode:
```

By default, it uses tape transport #0. You may select an alternate tape drive when entering tape mode:

```
Name of database: /t 4
Tape mode:
```

Tape Cache

Tape mode keeps the last 200 buffers in memory because the Britton Lee database server tape controller does not support backspacing a single block. The cache makes some things easy. For instance, to look at the last few blocks of a file, read until the next file mark. The blocks will be in memory and you may look around in it at random.
Whenever you request a page that might be either in the cache or on the tape, CKDB asks:

```
Look in cache?
```

If it does not find what it is looking for there, then it will ask

```
Read tape?
```

before it tries to read the next block from the tape.

Move Backwards A Block - b

This command does not back space the tape; it looks in the cache for the block preceding the current one. If the page is not in the cache, CKDB will print:

```
Tape command: b
Cannot read backwards.
Tape command:
```

and stay at the current block.

Display Current Block - d

Tape mode always has a "current block" which is the last block that it read or looked at. Displayed enters a version of "page dump mode" which is very similar to that of the "/d" command. You may print out the page as either tuples, octal bytes, or characters. Only system relations may be printed as tuples because CKDB cannot tell what the attributes of any users relations on the tape might be.
A few of the regular dump mode commands are not available in tape mode; specifically, the ones which would read pages from the disk (the l, f, and p).

Read Forward to a File Mark - f

Read forward does not use the "search to a file mark" command of the tape controller; it reads each block. This lets it put the blocks before the file mark into the tape cache so that you may look at them later.

Search for a Certain Page Header - h

You may search the cache and/or the tape for a particular page with the h command. It can find a block which has a certain pageid, nextpg, or relid.

List Blocks In Tape Cache

List blocks prints out the blocks in the tape cache with their block offset from the start of the tape and, if the verbose option is set, the pageid's in the headers of each page.

Go to Next Block - n

This is equivalent to:

<cr> to continue: .+1

Rewind the Tape - r

Rewinds the tape but does not wait for it to complete. A second rewind command will wait for the first one to finish and then report:

tape caution error #2: at load point

Seek to a Certain Block - s

Seek looks for a certain block number either in the cache or on the tape drive, or both. The number is a 2K block offset from the start of the tape.

Exit from CKDB /x

This command allows a Britton Lee database server to be rebooted from inside CKDB; instead of having to be near the machine. Before the reboot is attempted, CKDB will ask for confirmation:

Really reboot? y

The message

turn front panel switch to off please...

will appear on early production database processors in which this self-reset is not possible. In this case, it will be necessary to turn off the Britton Lee database server with the front panel keyswitch.

Errors

Introduction

There are many errors that CKDB detects. Some of them are so minor that CKDB can just note the inconsistency and continue. Others are so severe that CKDB cannot reliably determine what action to take next. On these it will stop the examination of the current object (database, relation, or index) and continue to the next one.
All diagnostic messages are in one of the following formats:

```
* * * Error in <relation name> <relid>
* * * Harmless Error in <relation name> <relid>
* * * Correctable Error in <relation name> <relid>
* * * Severe Error in <relation name> <relid>
* * * Fatal Error in <relation name> <relid>
```

and are followed by a description of the error.
CKDB groups together strings of related errors. For instance, if a relation has been deleted from the relation relation but has not had its disk space freed, CKDB could issue several dozen messages complaining about each tuple that is missing.
Instead it will print the first few errors and then note:

```
* * * 26 more errors
```

Mild Errors

CKDB attempts to deal with three classes of errors, a "plain" error, a "harmless" error, and a "correctable" error.

1. A "plain" error ("*** Error") is one which can easily be corrected by the user without any loss of data. This type of error is reported when an index on a relation has been found to be inconsistent. Simply recreating the index will usually take care of it.

2. A "harmless" error is another type of mild error. Harmless errors are less severe than plain errors but cannot always be easily corrected by user intervention.

3. A "correctable" error is one that CKDB knows is safe to fix. If the "fix" option is enabled, CKDB will correct it; if not it will print:

   ```
   Not corrected
   ```

   and continue checking the relation.

Severe Errors

Severe errors usually cannot be corrected as easily as milder errors. These are not necessarily worse than mild errors; they simply are not always correctable without losing any data. If the error is in a system relation, it may be necessary to load the database from the last dump.

Fatal Errors

A fatal error indicates that CKDB required some information that it could not obtain. The rest of the current relation or index will not be examined. It is possible for other errors to cause a more mild error to be upgraded to a more severe one.

Summarizing Errors

CKDB remembers which relations and indices have errors so that probiems may be summarized. Just as CKDB uses the unused "s1" column in blockalloc to verify disk allocation, some of the spare attributes in the relation and indices relations are used to record when and where errors are seen.

Remembering Errors of a relation

Errors are recorded in the previously unused attribute "s1" in the relation.

Bit	Value	Meaning
001		fatal error in this relation
002		relation has a bad index
010		CKDB completed its examination of this relation
020		some type of error
040		severe error in this relation

Remembering the Errors in an Index

Each bit may be set independently of the others.
A bad index is marked by setting bit 3 (octal value 010) bit in the attribute "stat" of the tuple.

Examination Sequence

Introduction

Under normal circumstances, CKDB does its exhaustive verification process on the whole database. By setting the -r, -b, or -i options, this basic action can be adjusted to suit your needs.

Open the Database

The first step is to open the database that you want to check. This is always done whether or not any of the above options are set.

Look for Incomplete Updates

Scan the transaction and back logs to determine whether there may be an incomplete transaction in the database. Partially executed transactions can leave the database in an inconsistent state and cause CKDB to report spurious errors. If it is not in a "safe" condition CKDB responds:

```
Recovery should be run on <database>
Go ahead anyway?
```

Enter y to check the database, or n to return to the top-level prompt.

Check Blockalloc

CKDB depends heavily on blockalloc; it stores status information about each page in the "s1" attribute. To avoid generating spurious errors, it does a quick check of blockalloc. Examine the system catalogues, blockalloc, and disk_usage. CKDB uses these relations to detect inconsistently allocated pages. It must be able to trust them (at least partially) before it can proceed with the verification.

Check a relation

For each relation:
Check the data pages of the relation for malformed tuples. Also, make sure that the pages are correctly allocated to the relation.
For each index:

1. Print the type and keys of the index.

2. Read the root page of the index, then read the level immediately underneath it; continue until CKDB gets to the level of the data pages. This is a horizontal scan of each level of the index following the sibling pointers of the tree.

3. Search the index recursively. Starting from the root, check the current page, then the lesser children, then the greater children. This section verifies that the tree structure of the index is correct and determines whether the information in the index agrees with the data of the relation.

4. Scan the index again, looking for pages which were found in the horizontal search of each level, but were not seen in the vertical recursive pass.

If any errors were found, record the type of problem in the "s1" attribute of this object's relation relation tuple.

End of Database Checks

Find Uselessly Allocated Pages

Look through blockalloc for incorrectly allocated disk blocks. Common (but harmless) errors are:

Page allocated to <relation> but not used

and

Page allocated to non-existent relation

Find Useless Tuples in System relations

Verify that the system catalogues are consistent with each other. Here CKDB looks for attributes that refer to non-existent relations and other similar errors.

Summarize Errors

CKDB now prints the name of each object which has an error and the type of error. A completed listing of the possible errors, each with a brief explanation, is given.

Error Messages

Message Format

The following error messages use a shorthand for specifying the type of data that will appear. This is the mapping of abbreviations to their meaning.
%d A decimal number is printed; it will never have a leading 0.
%o An octal number is printed; it will always have a leading 0.
%P Print a virtual block number as three octal bytes separated by <and>, as in:

<23 : 130 : 363>

%H Print out a disk block address as either its virtual number (ala %P) or its physical disk block address as in:

cyl %d Head %d sector %d

The context of the error message should make clear to which controller and drive it refers. If the block is in the bad block remapping area of a drive, then it is printed with the physical address because blocks in the remap area do not have a virtual block number.

Correctable Messages

The "fix" option (-f) enables you to take corrective action when a correctable error is detected. It is possible to run CKDB to see if any fixable errors have occurred, set -f and then run it again to have the corrections applied.

ERROR MESSAGE:

```
Allocation state bit of %P is incorrect
```

The allocation state bit (010 in blockalloc.mode) is used during the create index process to distinguish the "old" index and data pages from the new ones being sorted. After the create has finished, the ALSTATE bits of all the data pages should be the same as the equivalent bit (02000) in relation.status.
CKDB corrects this by setting the ALSTATE bit of an index or data page to the corresponding value in the relation or indices' relations. This may not be correct if it actually is the relation relation which is incorrect.

ERROR MESSAGE:

```
Tuple number is incorrect: %P offset %d has %o,
should be %o
```

This error is specific to the blockalloc relation because its tuples never move about on a page. Thus, they always have the same structure; a simple array of 6-byte records in ascending order of tuple number.

<offset>	tno	mode	check	stat	relid
14	0	4	0	0	11
20	01	4	0	0	1
26	02	4	0	0	2
32	03	4	0	0	3
.					
.					
152	027				0
158	030				0
164	031				0

Table 1-2: Blockalloc tuples

The blockalloc tuple at offset 158 must always be tuple number 030 no matter what the configuration of the drive or how it was formatted.
Correction Action:
The tno of the tuple is changed to be its "index" into the array of blockalloc tuples that should be on the page.

ERROR MESSAGE:

```
Relstat says it is one-zone, but it isn't
```

The one-zone bit (040000) in relation.status says that all of the data and index pages on the relation or file are in the same zone. However, while scanning the relation, CKDB reads pages of the object that were not in that zone.
The only effect of this error is that if the object is destroyed then some pages will not be deallocated.

Corrective Action:
It is corrected by turning off the one-zone bit. This will increase the time it takes to destroy this relation, but it will correctly free all of its pages.

ERROR MESSAGE:

```
Lowzone is %P, should be %P
Highgrove is %P, should be %P
```

The allocation code maintains the lowest and highest zones which are used by an object. If the zone ranges are incorrect when the object is destroyed, the pages outside the known range will not be deallocated and would not be re-used.
If CKDB reads a page which is lower than relation.lowzone, or higher than relation.highzone, it can fix the error by extending the range as appropriate.

ERROR MESSAGE:

```
Root %P has a next page
```

The root of an index cannot have a "horizontal" page after it. If it does, it should not really be in the root page; there should be another level above it that points to both pages. The corrective action cleats the next page pointer. The index should really be remade.

ERROR MESSAGE:

```
Last data page %P thinks it has an overflow
page
```

The last data page of a relation has the "has-overflow" bit (04) set in its page status. This bit implies that there are other pages after the current one that have the same clustered index key values. You can correct it by turning off the "has-overflow" bit.

ERROR MESSAGE:

```
Lastpage in rel-rel wrong; is %P should be %P
```

The value in relation.lastpg is incorrect. When CKDB reads this page, it noticed that it was not at the end of data page list but there were more pages after it. Relation.lastpg only matter for relations that do not have a clustered index. Appends to "heaps" are made faster by maintaining this pointer to the end of the data pages so that new pages may be added quickly. CKDB fixes this by setting relation.lastpg to the last data page.

ERROR MESSAGE:

```
Firstpage in rel-rel wrong; is %P should be %P
```

The first data page that CKDB found by going through the clustered index was not the page specified in relation.firstpg. It is fixed by setting relation.firstpg to the first data page accessible through the index.

ERROR MESSAGE:

```
Firstpage in rel-rel wrong; is %P should be %P
```

The first data page that CKDB found by going through the clustered index was not the page specified in relation.firstpg. It is fixed by setting relation.firstpg to the first data page accessible through the index.

ERROR MESSAGE:

```
%P forgot about its overflow page %P
```

The first page does not have the has-overflow bit on, but the second page does have the is-overflow bit. This is detected while checking the clustered index. CKDB fixes its setting the has-overflow bit in the first page's status.

ERROR MESSAGE:

```
Extra index tuples after %d on %P point
after last data %P
```

This error occurs during the recursive scan of an index. CKDB has seen the last of the data pages, but there are still more index tuples to be looked at. The data pages that the "extra" index tuples point to are linked back into the main data page list. If other errors still show up, then retrieve the data through the index and recreate the relation.

ERROR MESSAGE:

```
%P allocated to non-existent relid %d
```

The indicated page is allocated to a relation which no longer exists. The page may be deallocated. If there were any errors in the system catalogues, CKDB will ask for confirmation before freeing the page. A system error could make CKDB not find an object that exists.

ERROR MESSAGE:

```
%P allocated to %.12s but not used
```

The indicated page is allocated to the relation or file but CKDB did not have to read the page during its examination of the object.

ERROR MESSAGE:

```
%P refers missing tuple %o
```

The tuple number table on the specified page says there is a tuple of a certain number on the page, but CKDB did not find the tuple during its scan of the page. This is fixed by zeroing out the tuple number table entry, effectively completing the deletion of the tuple.

ERROR MESSAGE:

```
Nexttno of %P is %o, should be %o
```

The size of the tuple number table on the given page is incorrect. The correct value is the second number given.

ERROR MESSAGE:

```
Tnofree bit of %P is %o, should be %o
```

The "tnofree" bit in the page status is wrong. It either says that there is a free slot in the tuple number table though the table is full or it says that the table is full when there remains an unused tuple number. It is corrected by making past agree with the tuple number table.

ERROR MESSAGE:

```
%H is free, should be marked bad
```

A bad block was encountered during the scan disks mode ("/s") that was marked free in blockalloc. This can only occur on disks that were formatted with DFU before release 30, i.e., disks without a remap area.

ERROR MESSAGE:

```
%P allocated to non-existent database %d
```

During the scan disks mode ("s) CKDB found a zone that was allocated to a non-existent database which used to have the given dbid. This may be caused (in a multi-disk system) by:

1. taking a drive off-line
2. destroying a database that was on the drive
3. putting the drive back on-line. It is fixed by freeing the zone.

ERROR MESSAGE:

```
%P says tuple %o is at %d, should be %d
```

There are two ways to find a tuple on a page. If you have tid, then the tuple number indices into the tuple number table which is on each data page. This gives the offset from the start of the page to the tuple. The other way to find a tuple is to start at the beginning of the page and do either linear or a binary search down the page until you find it. This error means that the tuple number table says the tuple is at the first given offset, but the tuple is actually at the second one. It is more likely for the second offset to be correct.

Harmless Error Messages

ERROR MESSAGE:

```
Small tuples were not found on %P,
but it should have them
```

The header of each page contains a copy of the relation.status bit which says whether a relation has small tuples or not (value of 020). The page header's small tuple flag did not agree with the one in the relation relation.

ERROR MESSAGE:

```
Small tuples were found on %P<, but
it should not have them
```

This is a harmless error if the tuples are of fixed length; however, it is severe if the relation has any compressed attributes.

To fix:

It may be corrected by copying the small-tuple bit value from the relation relation to the page.

However, if this error seems to be occurring on every data page, it is usually the relation tuple that is wrong. You can calculate the correct value for the bit by looking at relations.maxwidth or the appropriate tuples attribute and summing up the sizes.

ERROR MESSAGE:

```
Tuple %T refers to non-existent relid %d
```

At the end of a full database scan neither the -r nor the -b option were specified), CKDB looks at the tuples in the attribute, indices, protect, query, crossref, and descriptions relation. If it finds any tuples which contain a relid which is not in the relation relation it prints this message.

ERROR MESSAGE:

```
%P allocated to %.12s but not used
```

The page is allocated to the indicated relid, but CKDB did not have to fetch the page during its examination of the relation. This is harmless because the only side-effect is that it wastes space; besides this, the relation is consistent.

ERROR MESSAGE:

```
Index page %P has tnofree set
```

The "tnofree" bit was set in the page header of an index page. This bit is only meaningful on data pages, so it is an interesting, if harmless, condition if it is set on an index page.

ERROR MESSAGE:

```
%H is allocated in blockalloc but
not used by the disk
```

A blockalloc tuple in the remap region of a disk says that the page is allocated for remapping, but the physical disk header itself indicates that it is not currently being used. This could be changed to a correctable error if it happens frequently enough that it is bothersome. The main undesirable side-effect is that the page is wasted and cannot be used if a block goes bad in the "normal" region of the disk.

ERROR MESSAGE:
```
It should be refined
```

The size of the query buffer can be changed by a "K" tuple in the configure relation. A stored command or stored program cannot run if the query buffer at the time of execution is smaller than the query buffer at the time of the query is defined. If you tried to use the indicated query, the Britton Lee database server would respond with error 192 (redefine the stored command/program).
Views are not dependent upon the size of the query buffer; they do not become out-of-date.

ERROR MESSAGE:
```
%H is not remapped to %H
```

On a remap zone disk, a bad page points to a good page in the remap area. In addition, the good page points back to the bad page in the normal region of the disk; they both point to each other. This helps the system manage the remapping information. CKDB prints this message if the good page does not point back to the bad page. This can happen if a page in the remap zone is mapped out, either automatically (on a mirrored disks system) or by hand (with DFU).

User Correctable Errors

These problems can be corrected by IDL, SQL or copy commands. Usually no data is lost by one of these errors. Many are fixed by one of the following procedures:

Remake Indices

Create the clustered index without destroying it first. This destroys any non-clustered indices as well. You will have to create them again.

Remake Index

Destroy the index specified in the error message and create it again. It is not sufficient to simply recreate the index with the "with recreate" option.

Copy

Copy out the relation, destroy it, and copy it back in again. Stored commands, stored programs, and views defined on the relation will have to be destroyed and re-created.

Copy Through Index

A simple copyout will not work. Since it follows the data pages list, it will not retrieve the data which is accessible only through the index. Do a retrieve, but send the appropriate PLAN options so that the index specified in the error message will be used.

ERROR MESSAGE:
```
Tuple %T is inconsistent mode %o relid %d
```

When checking the blockalloc relation at the start of a database examination, CKDB found an allocation tuple which contains inconsistent information. Either the mode attribute said that the page was free and the relid said that the page was used, or the mode said the page was used and relid was zero, indicating that the page should be free.

ERROR MESSAGE:

```
Index page %P is linked vertically
but not horizontally
```

The index page indicated is not correctly lined into the b–tree structure of the index. An index has child pointers which point down in the tree, towards the data pages; brethren pointers point sideways to other index pages at the same level above the data pages. CKDB detected a page that was only accessible via the horizontal pointers, but not the vertical pointers or vice versa.

ERROR MESSAGE:

```
%P was linked horizontally but not vertically
```

Remake Indices

ERROR MESSAGE:

```
[Data|Index|File] page %P is referenced
more than once
```

The page is used in two different contexts in the relation. It might be used as both an index and a data page.

Correct by:
index
Remake Indices
data or file
Probably at least a page's worth of data has been lost. COPY

ERROR MESSAGE:

```
[Data|Index|File] page %P is from a file
```

The "file data page" (0100) bit should be on in blockalloc.mode if the page is a file data page. A page belonging to a normal relation, a transaction log, or the clustered index on a file should never have this bit set. It is most likely that the bit was accidentally turned on in blockalloc.

Correct by:
index
Remake Indices
data or file
data COPY

ERROR MESSAGE:

```
[Data|Index|File] page %P is both an
index and a data page
```

The page is used both as a data page and as part of an index.

COPY

ERROR MESSAGE:

```
Data page %P is not in data list
```

CKDB could get to the page only through an index, though all data pages should be in the data pages list. The index points to more tuples than the data.

COPY THROUGH Index

ERROR MESSAGE:

```
Disk usage tuple %D to %D is
a subset of the fragment from %D to %D
```

The tuples in disk_usage should correspond to the range of sequential blockalloc pages. This error means that disk_usage and blockalloc

disagree on the size of this particular fragment. Disk_usage thinks that it is smaller than it actually is in blockalloc. A dump of this database might not be loadable due to the conflicting information.

To Fix:

Correct by:? (This probably should be a severe error.)

ERROR MESSAGE:

```
Found %D data tuples in index;
should have %D tuples
```

The number of tuples retrievable through the index was not the same as the number that CKDB saw when it was searching the data pages.

Correct by:

If the index finds more tuples that are in the data, you should have gotten at least one error message "Data page %P not in data page list". That is the actual error.

If the index has fewer tuples, Remake Indices.

ERROR MESSAGE:

```
Lesser @%d in %P is larger than first key
in child index page%P
```

The index is no longer sorted correctly.
Remake Index

ERROR MESSAGE:

```
Greater key @%d on %P is smaller than last key
in child index page%P
```

The index is no longer sorted correctly.
Remake Index

ERROR MESSAGE:

```
Index tuple @%d on %P points to overflow page
%P
```

Clustered index tuples should only point to main (not overflow) data pages. The index is not consistent and will not retrieve all the tuples. This can only occur on a clustered index.

Remake Indices

ERROR MESSAGE:

```
Index tuple @%d on %P points to %P
```

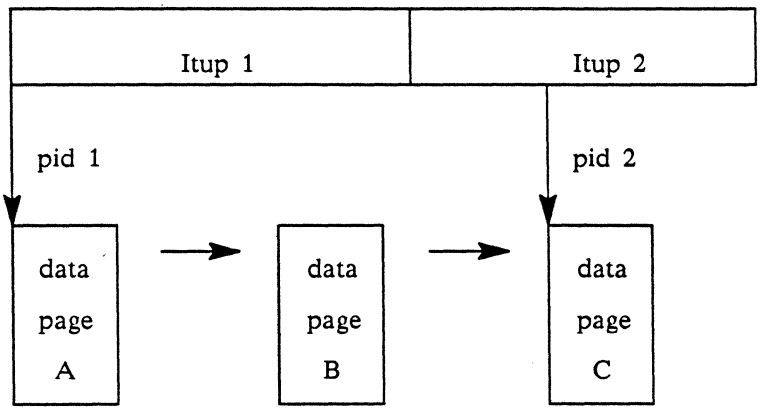

Table 1-3: Clustered Index Pages

When CKDB reads "i1", it remembers the data page that the next index tuple "i2" should point to. In this case, A says i2 should point to B. However, i2 actually points to C.

Clustered index only
Remake Indices

ERROR MESSAGE:
`Data %P thinks %P pstat %o is overflow`

The first page has the P_HASOVER bit (04) on in its page status. Such a page should always be followed by a P_ISOVER (010) page which may also have the P_HASOVER bit set.

Clustered index only
Remake Indices

ERROR MESSAGE:
`Extra data page %P after end of clustered index`

The last index tuple should point to the last data page. However, the next page pointer in the data page header pointed to another main data page.

Clustered index only
Remake Indices

ERROR MESSAGE:
`Index tuple @%d on %P points out a data page to data tuple`

The index tuple points to a data page that was not seen when CKDB read the data pages list. This probably means that there are tuples missing from the data pages which are only accessible through a non-clustered index.

Non-clustered index only
COPY THROUGH Index

ERROR MESSAGE:
`Index tuple @%d on %P has no data tuple %T`

Each non-clustered index tuple should point to a data tuple which has the same attribute values as the key fields of the index. CKDB could no find the desired tuple; there was no tuple with that tuple number on that data page.

Non-clustered index only
Remake Index

ERROR MESSAGE:
`Index tuple @%d on %P does no match its data tuple %T`

Each non-clustered index tuple should point to a data tuple that has the same attribute values as the key fields of the index. This error means that the data tuple did exist but its values did not match.

Non-clustered index only
Remake Index

ERROR MESSAGE:
`Rel-rel says maximum relation width is %d; should be %d`

The maximum width in relation.maxwidth is not the same as the width computed by adding up the space needed by each attribute of the relation.
COPY

ERROR MESSAGE:
```
Rel-rel says maximum relation width is %d; this
is larger than IDM maximum %d
```

Somehow a relation was created which has a maximum tuple size which is larger than the limit that the Britton Lee database server supports.

COPY the relation, but either split the relation up into two or more relations or reduce the lengths of compressed attributes.

ERROR MESSAGE:
```
Rel-rel says it has [big|small] tuples
```

One of the bits in relation.status (020) indicates whether or not the maximum tuple width is less than 256 bytes or not. If this bit does not agree with the value of maxwidth that CKDB computed, this error is printed.

COPY

ERROR MESSAGE:
```
Rel-rel says tuplen is %d; should be %d
```

The field relation.tuplen was not the same as the value that CKDB computed using the information in the attribute relation.

COPY

ERROR MESSAGE:
```
Empty index page %P on level %d
```

There are not supposed to be any empty index pages in the upper levels of an index.

ERROR MESSAGE:
```
Duplicate keys in unique/clustered index on %P
```

Index tuples which contained the same key values were found in a clustered or unique non-clustered index. Duplicate keys are only allowed in non-unique, non-clustered indices. This error is not an error in the data tuples; it is complaining about the actual index tuples that make up the B-tree structure of the index.

Remake Index

ERROR MESSAGE:
```
Identical tuples on %P
```

A pair of completely identical (including tids) index tuples were found. The offending tuples are printed after the error message.

Remake Index

ERROR MESSAGE:
```
Tids of identical keys out of order on %P
```

In a non-unique, non-clustered index, tuples which have identical key values should be sorted according to the tid of the data to which the index tuples refer.

Non-clustered only
Remake Index

ERROR MESSAGE:
```
Index keys out of order on %P
```

The tuples of an index are always supposed to be sorted according to its keys. The offending tuples are printed after the error message.

ERROR MESSAGE:
```
Freeoff points in middle of last tuple @%d on
%P
```

The header of the indicated index page is incorrect. It says that the end of the useful index tuple space on the page is actually in the middle of the last index tuple.

Remake Index

ERROR MESSAGE:
```
Data tuples @%d and @%d on %P have the same
keys
```

A relation which has a unique clustered index has two data tuples with the same keys. This is a violation of the uniqueness criterion.

Remake Indices

ERROR MESSAGE:
```
Duplicate tuples @%d and @%d on %P
```

CKDB found two adjacent identical data tuples in a relation which has a clustered index.

Remake Indices

ERROR MESSAGE:
```
Data tuples out of order @%d and @%d on %P
```

Data tuples are supposed to be sorted according to the keys of the clustered index, if any. The Britton Lee database server may not be able to find tuples that are improperly positioned.

Remake Indices

ERROR MESSAGE:
```
Tuple @%d on overflow page %P does not match
previous tuple @%d on %P
```

All tuples on overflow pages must have the same values in their key fields. These attributes are the key fields of the clustered index.

Remake Indices

ERROR MESSAGE:
```
Found data tuple @%d on %P which is smaller
than index tup @%d on %P
```

The parent index tuple points to a child data tuple which is smaller than it. The data tuple may not be found when searching through that index.

Remake Index

ERROR MESSAGE:
```
Found data tuple @%d on %P which is larger than
index tup @%d on %P
```

The tuple after the parent index tuple points to a child tuple which is larger than it. The data tuple may not be found when searching through that index.

Remake Index

ERROR MESSAGE:
`Leaf & data set in %P`

Two bits were set in the indicated page's header that are never supposed to be on at the same time. The leaf bit says the page is in the bottom page of a non-clustered index. The data bit says the page is in the data pages list of the relation.

Remake Indices

ERROR MESSAGE:
`%P is not linked into %.12s`

This message is only printed by the disks scanning function of CKDB (/s). It means that the indicated zone thinks it is allocated to the particular database, but the database's disk_usage relation does not contain that zone.

ERROR MESSAGE:
`%P on level %d, should be %d`

CKDB found the indicated page to be on the first numbered level,but it should have been on the second given level. Data pages are always level zero; index page levels vary from 0 to 10. This error can be encountered almost any place in CKDB.

ERROR MESSAGE:
`Index page %P links level %d to %d`

One of the horizontal index page pointers points to a page that is on a different level.

Remake Indices

ERROR MESSAGE:
`Index status bit %o not compatible with index %d`

There are two ways that identify an index as being clustered or non-clustered. One is the 02 bit in the indices.stat attribute, which says that it is a clustered index. Also, clustered indices always have an index id of 0. This error occurs when the status bit disagrees with the index id; e.g. the bit is 2 but the index id is non-zero.

Remake Indices

Severe Error Messages

ERROR MESSAGE:
`relation tuple expected it not to have a [non]clustered index, but one was found`

A pair of bits in relation.status indicated whether the relation has any clustered (02) or non-clustered (04) indices. These errors mean that the indices that should have been there were not found in the indices relation or that there were unexpected indices that relation.status did not mention.

ERROR MESSAGE:

```
Relation tuple expected it to have a
[non]clustered index but one was not found
```

To Fix:

Destroy the indices or COPY

ERROR MESSAGE:

```
Small tuples were not found on %P, but it
should have them
```

The header of each page contains a copy of the relation.status bit which says whether a relation has small tuples or not (value of 020).
The page header's small tuple flag did not agree with the one in the relation relation. It is corrected by copying the relation relation value to the page. If all or many of the page headers are diagnosed as being wrong, it probably is the relation relation which is incorrect.

ERROR MESSAGE:

```
%P allocated to %d
```

The indicated page is allocated to the relid given in the message and not to the relation that is using the page. This check is performed before the page is actually read. It may or may not contain the correct relid.

ERROR MESSAGE:

```
Allocation error:%P is bad, mode %o
```

The blockalloc tuple of the indicated page says it is a bad page. This means that the page probably is not reliably readable, so not one should try to use it.

ERROR MESSAGE:

```
Allocation error:%P is free, mode %o
```

The blockalloc tuple of the indicated page shows it is not allocated to any relation. Thus, a process which wanted another page might take this one, since it says it is available. Once this page is used again, a "syserr:bufrd" might occur.

ERROR MESSAGE:

```
Allocation error:%P is not used, mode %o
```

The blockalloc tuple of the indicated page says it is not used. The page is neither free not used, nor involved in remapping. This condition is illegal whether or not the page is being used by a relation.

ERROR MESSAGE:

```
Allocation error:%P is (used) -remapped, mode
%o
```

The blockalloc tuple of the indicated page says it is an alternate page used in remapping. However, there are not supposed to be any pointers to the alternate "target" page, only to the original bad page.

ERROR MESSAGE:

```
Allocation error:%P is demanded, mode %o
```

The blockalloc tuple of the indicated page says it has been "hard" allocated. This is the "demand" allocation state that has not been implemented in the Britton Lee database server.

ERROR MESSAGE:

```
File page %P is allocated to another relation.
```

A data page from a file is allocated to another relation.

ERROR MESSAGE:

```
File page %P is not allocated as file data
```

Each data page of a file should be allocated in blockalloc. The mode attribute should contain the file data page (0100) bit.

ERROR MESSAGE:

```
Disk_usage tuple low %D high %D does not match
blockalloc
```

Before examining a database, CKDB reads blockalloc and constructs a fragment table. It should correspond to tuples in disk_usage with relid = 32768.

ERROR MESSAGE:

```
[Data|Index] tuple %T has an unsorted attribute
table
```

A relation with compressed fields has an attribute offset table at the beginning of each tuple. It contains the position and lengths of each of the variable length fields. The Britton Lee database server requires that this table be sorted in nondescending order.

ERROR MESSAGE:

```
Freeoff %d on %P
```

Freeoff is one of the elements of the page header. It contains the offset from the start of the page to the end of the used tuple space on the page. This error occurs when the offset points off the page; i.e. it is either less than the size of the page header or greater than the size of the page. It probably should be a fatal error.

ERROR MESSAGE:

```
Data area of %P extends into tuple table
freeoff %d
```

Freeoff marks the end of the used tuple space on a page. This error means that the two ways of finding the amount of used space on a page disagree.

ERROR MESSAGE:

```
Tuple %o is used both @%d and @%d on %P
```

Each data tuple on a Britton Lee database server is uniquely identified by the combination of the page number that it is on and its tuple number on that page. The indicated page had two different tuples which claimed to have the same tuple number. CKDB cannot determine which one, if either, is correct, so it skips the rest of the relation.

ERROR MESSAGE:

```
Size of data tuple @%d on %P is %d
```

The indicated data tuple had an illegal size; either negative or larger than the largest allowable tuple size for the relation (relation.maxwidth).

ERROR MESSAGE:

```
%P contains a tuple @%d marked as deleted
```

The tuple space of a page and the tuple number table of the page disagree on which tuples are present. The tuple number table has indicated deleted tuple which is still present on the page.

ERROR MESSAGE:
```
%P has tuple %o @%d >=nexttno %o
```

The tuple space of a page contains a tuple number which is higher than the largest tuple number that the page header knows about.

ERROR MESSAGE:
```
%P has large index tuples
```

An index page always contains small (<256 byte) tuples. The indicated page header did not have the small-tuple bit on.

ERROR MESSAGE:
```
Can't read disk header of %H
```

CKDB could not read the physical sector header of the indicated cylinder/head/sector. The IDM/RDBMS code needs to be able to read the physical sector header of every page on a drive, thought the actual data portion of a page may be unreadable.

ERROR MESSAGE:
```
%H is remapped on the disk but not in
blockalloc
```

During scan disks mode (/s), CKDB found that the physical sector header says the page is remapped but blockalloc does not know about this. This can cause the page to appear to be free and be re-allocated which may eventually generate a "syserr:bufrd".

ERROR MESSAGE:
```
%H is not mapped into the reserved region but
to %H
```

During scan disks mode (/s), a page was found to be incorrectly remapped. The page might seem to be "free" and could be re-used by another relation. It is also possible that load database could fail. This error only occurs on disks formatted by release 30 DFU or later.

ERROR MESSAGE:
```
%H  badly mapped into remap region at %H
```

During scan disks mode (/s), the remap pointer in the remap zone points to a bad page (in the virtual region of the drive) which does not point back to the remap some page. (A points to B but B does not point to A, though A does point into the remap region.) This error only occurs on disks formatted by release 30 DFU or later.

ERROR MESSAGE:
```
Can't read blockalloc page %H
```

The indicated page has a hard disk error. CKDB cannot check the pages in this zone.

ERROR MESSAGE:
```
Expected relid 11 on %H but found relid %d
```

CKDB could read a blockalloc page but it did not have the correct relid on it.

ERROR MESSAGE:
```
Can't read disk header for remap page %H
```

The physical sector header of the page was not readable.

ERROR MESSAGE:

```
%H is remapped into the remap region
```

A page in the remap region of a drive is remapped to another page in the remap area. This is not allowed; bad page in the remap area are just marked bad and are not used.

ERROR MESSAGE:

```
%H is remapped off the drive
```

The physical sector header of the page contains an alternate cylinder address which is larger than the number of cylinders on the drive.

ERROR MESSAGE:

```
%H erroneously says %H is bad
```

The first page (which is in the remap zone) points to the second page in the virtual area of the disk, but the second page is not remapped at all.

ERROR MESSAGE:

```
Blockalloc thinks that %P is remapped
```

The blockalloc tuple of a page thinks it is remapped (mode has the 020 (bad) and 0200 (remapped) bits on), but the physical sector header says the page is not.

ERROR MESSAGE:

```
%P mode of bad & remapped on a remap-zone drive
```

Drives that have been formatted with release 30 or greater DFU should have neither the bad (020) nor the remap (0200) bits on any page's blockalloc tuple.

ERROR MESSAGE:

```
%P remapped from database %d to %d
```

On a drive that was formatted with a version of DFU before release 30, a page was remapped from one database to another. Only blockalloc is allowed to be remapped across databases.

ERROR MESSAGE:

```
%P remapped to %P which isn't used and remapped
```

On a drive that was formatted with a version of DFU before release 30, a page was remapped to another page but the blockalloc tuple of the second page did not say that it was used in remapping.

ERROR MESSAGE:

```
%P remapped to relid %d on %P
```

On a drive that was formatted with a version of DFU before release 30, a page was remapped to another page which was allocated to a different relation. Both the original (bad) page and the alternate (good) page should have the same relid in their corresponding blockalloc tuples.

Fatal Error Messages

ERROR MESSAGE:

```
No blockalloc tuple for %P
```

The blockalloc tuple for the indicated page did not exist on the page that should have contained it. This usually means that the page was trashed.

ERROR MESSAGE:

```
%P belongs to %.12s dbid %d
```

The page did not belong to the database that CKDB was checking. The zone in which the indicated page belongs is allocated to the given dbid. A data base id of zero means that the zone is free.

ERROR MESSAGE:
```
%P no in blockalloc and disk_usage
```

The indicated blockalloc page said that it belongs to the correct database but neither disk_usage nor the real blockalloc admit to knowing about it.

ERROR MESSAGE:
```
Loop @ %P
```

CKDB discovered a loop in the linked list of data pages for blockalloc. Because CKDB needs to use blockalloc for recording state information about each page, it cannot continue the examination of the database. Prior to the error message, CKDB prints out the 50 fragments it already found.

ERROR MESSAGE:
```
%P remapped to non-existent page %P
```

The first page in the message is allegedly remapped to the second page which is not on-line. This is a fatal error because pages can only be remapped to a different cylinder/head/sector combination on the same drive. For this error to occur, the cylinder number in the physical disk sector header must be larger than the number of cylinders on the drive.

ERROR MESSAGE:
```
%P in blockalloc remapped to %P relid %d
```

A blockalloc page is remapped to a page which is not allocated to blockalloc but to relid %d.
This message occurs when the page is remapped to another database which is legal only for blockalloc pages.

ERROR MESSAGE:
```
Loop in data pages:%P points to %P
```

One of the page in the data pages list of a relation points back to an earlier page in the list. CKDB can do nothing more with this relation.

ERROR MESSAGE:
```
Loop in rel-rel
```

There were more than 32,767 tuples in the relation relation; CKDB diagnoses this as a loop in its data pages.

ERROR MESSAGE:
```
It has no attributes
```

A relation or log has no tuples in the attribute relation. CKDB can do nothing with it.

ERROR MESSAGE:
```
No root page for nonclustered index
```

It is possible for clustered indices to not have a root index page if there is still one data page. However, non-clustered indices must always have a root page. CKDB skips this index.

ERROR MESSAGE:
```
Empty page %P
```

An empty page was found. This is a fatal error because there are no child page pointers to follow. CKDB can go no further. There is another message similar to this one in sort.c. CKDB skips the rest of this index.

ERROR MESSAGE:
```
It has no query text
```

Stored commands, stored programs, and views do not have any data in pages on disk. Their contents are stored in the system query relation. Attempting to run the query will result in a syserr "readqry: empty cmd" in the main IDM/RDBMS code. Destroy the object and redefine it so that the possible syserr will be prevented.

ERROR MESSAGE:
```
Data page %P points to index page %P
```

While reading the data pages list, CKDB found an index page. The variant form of this message means that the first page of the relation (relation.firstpg) was wrong. CKDB can do nothing more with this relation.

ERROR MESSAGE:
```
%P is not a data page
```

CKDB was verifying the clustered index and the data pages of a relation and found an index page where it expected to find a data page. This means that was supposed to be the bottom level of the index actually turned out to have another index page below it. This is caused by a random child pointer that happens to point into one of the indices of the relation.

ERROR MESSAGE:
```
Found data page %P in index
```

While reading the pages of an index, CKDB found a data page.

ERROR MESSAGE:
```
Can't read blockalloc page %P
```

Dump page mode (/d) tries to find the blockalloc tuple of each page that is read so that it can print out any interesting information about it. This message is printed if there is a hard disk error on the page or if the page is so trashed that the appropriate blockalloc tuple cannot be found. The original page requested will still be read, though its allocation information will not be available.

ERROR MESSAGE:
```
More than %d attributes
```

A relation had more than 250 tuples in the attribute relation. Since 250 is the maximum number of attributes a relation may have, CKDB does not handle storing any more.

ERROR MESSAGE:
```
No attribute %d
```

CKDB could not find the indicated attid in its memory array of attributes for the current relation.

ERROR MESSAGE:
```
No more descriptors
```

CKDB wanted to open another relation but it would not because it had too many already open. This may be caused by excessive nesting of commands, such as running /r from inside a /q from inside /d under

certain exceptional error conditions. It does not imply a fatal error in the database; just reboot and run CKDB again.

ERROR MESSAGE:
```
Can't continue
```

In several places CKDB scans all of blockalloc to load for certain information. It cannot do this if there is a fatal error in blockalloc.

ERROR MESSAGE:
```
%P is remapped to a non-existent block
```

The blockalloc tuple of the page to which the indicated page is remapped cannot be found. Either an unreadable or trashed blockalloc page may be the cause. It is also possible that the page is remapped off-the drive which must be caused by an excessively large cylinder number in the physical block's remap header.

ERROR MESSAGE:
```
Asked for %P, go %P
```

The first page number was read, but the data had the page number of the second page. This can be caused by a remapping error or by a page being written out to the wrong location.

ERROR MESSAGE:
```
%P has wrong relid %d
```

The indicated page should have belonged to the current relation, but the relid on the page says that it belongs to another. The IDM/RDBMS code would get syserr:bufrd if it tried to read this page.

ERROR MESSAGE:
```
Tid %T in an index
```

A tid (which can only identify a data tuple) pointed to an index range.

ERROR MESSAGE:
```
Bad tno %o, pg %P
```

The tuple number component of a tid was larger than the maximum valid tuple number for the indicated page.

ERROR MESSAGE:
```
Bad offset %d, tno % opg %P
```

The tuple number table of the indicated tid pointed off the page.

ERROR MESSAGE:
```
Index tuple @%d on %P has size %d
```

The indicated index tuple is smaller than the minimum allowed for that index.

ERROR MESSAGE:
```
Data tuple @%d on %P has size %d
```

The indicated data tuple is smaller than the minimum allowed for that relation.

ERROR MESSAGE:
```
Data tuple @%d on %P has size %d > max size %d
```

The indicated data tuple is larger than the maximum allowed for that relation (relation.maxwidth).

ERROR MESSAGE:
```
%P not on-line
```

CKDB wanted to read the page, but there are no accessible disks which contain it.

ERROR MESSAGE:
```
tnochk:tno %o on %P
```

The tuple number component of a tid was larger than the maximum valid tuple number for the indicated page.

ERROR MESSAGE:
```
tnochk:tno %o on %P
```

The tuple number table of the indicated tid pointed off the page.

ERROR MESSAGE:
```
tuple extends beyond end of page
```

While looking at a page in either /d or /t mode, CKDB found a tuple which was smaller than the maximum size of tuples for the relation but which started so far into the page that it overflowed onto the next page.

Disk Formatting Utility (DFU)

Introduction

The Disk Formatting Utility (DFU) lets a user format a system disk and up to 15 user disks. DFU is also used to create, recreate, or destroy a system database on any disk. Disks and their databases, newly attached to an existing Britton Lee database server, may be merged into the system. DFU can validate the system database, display a list of all disks on-line, unformat disks from the system, modify the configure relation, and mapout and remap blocks without reformatting.
When a mirror Database Processor (DBP) is in a Britton Lee database server, disks may be paired as mirrors of each other. This applies equally to system and user disks.

Disk Organization

A Britton Lee database server can be configured with 1 to 4 disk controllers (depending on the model).
Each controller supports up to four disk drives. One disk, and only one disk per Britton Lee database server, must be specified as the system disk, containing the system database.
The system database contains relations with system level information and files that contain the IDM/RDBMS software. User databases may occupy the remaining space on the system disk. Because of the type of information stored on it, the system disk receives special attention during the formatting process.
Formatting a disk also tests its ability to record data properly and ensures that all operations on the disk will access usable blocks. Formatting each track consists of the following steps:

- Write a known test pattern onto each block of a given track of the disk.
- Read the data from each of the blocks on the track and compare it to the known pattern.
- If the data on the block does not match the known pattern, then the block is marked as bad.

An alternate block in a different area of the disk is designated to replace the bad block. This is known as remapping bad blocks, and is transparent to the user. From this point on, the disk controller will access the alternate block whenever an attempt is made to access the original bad block.
DFU divides each disk of cylinders, tracks, and sectors into zones containing up to 254 blocks of 2048 bytes each. The exact number of blocks per zone is optimized for the particular disk being formatted. The first block in each zone is called the blockalloc and is part of the blockalloc relation. The blockalloc relation contains information on the status of the other blocks in the zone, such as:

- What database owns this zone
- Blocks assigned to each relation
- Blocks that are allocated, free, or bad

After all blocks of a zone are tested, the program writes the blockalloc for the zone in the first good block of each zone. The above steps are repeated for each zone of the disk. The blocks of the disk are assigned numbers. These VIRTUAL block numbers must be unique throughout the system. When initially configuring a Britton Lee database server, the system disk is assigned block numbers 1 through J, since it is always the

first disk to be formatted. However, the system disk can be any range of numbers, not necessarily starting with 1. Additional disks for user databases are assigned block numbers J+1 through K, K+1 through M, and so on.
The lowest and highest block numbers assigned to a disk are recorded in its master block at formatting time. The master block is the second usable block on the disk, normally cylinder 0, track 0, sector 1. But if sector 0 is bad, then sector 1 is assigned to blockalloc and sector 2 is the master block.
A disk's block numbers must be unique, and the range of each disk's block numbers must not overlap the block number range of any other disk. While this is ensured during initial system configuration, it can become a problem if a user replaces a disk in the system with one of larger capacity. The block number gap left by the removed disk will not be large enough to accommodate the range of block numbers needed by the new, larger disk.
To prevent block number overlap between the new large disk and the other disks in the system, the new disk must be assigned block numbers greater than any previously assigned. If this causes the block numbers of the new disk to be greater than the maximum number of blocks allowable on the Britton Lee database server ($2^{24}-1$) the user must remove other disks with the Unformat Mode described below, to create enough space for the new disk.
Each disk must also be given a unique name of up to twelve characters. This allows the Britton Lee database server to keep a separate entry for each disk in the data dictionary relation: *Disks*.

CAUTION

Do not use two-word disk names. Delineating marks, such as the spacebar or a blank ( ), a period (.), or an underscore (_) may cause SYSERRs or loss of disk recognition.

DFU protects against the intentional or unintentional destruction of data on the system disk because of the importance of its contents. The user may not directly reformat or unformat a system disk. To reformat the system disk, you must first move the system to another disk using the SysDestroy and SysCreate modes. Then Unformat the old system disk and Merge the system.
Much of DFU is intended to speed recovery from the loss of the system database. This can result from loss of the system disk or media failure on the system disk. All DFU modes are described in the DFU examples in this section.

Usage

A Britton Lee database server must be in Maintenance Mode to run DFU. The Britton Lee database server reads input from and writes output to the console port in this mode. A console terminal is normally connected to the console port. Alternatively, a host can interface the Britton Lee database server console port to one of its terminals.

Starting DFU

Refer to the *Installation* manual for diskette and tape loading and starting processes.

```
<BL series> - Filename: kernal dfu<RETURN>
<BL series> - Filename: load kernal dfu<RETURN>
<BL series> - Filename: loadtape kernal dfu<RETURN>
```

For all cases, the following response is a reasonable "facsimile" of what you will see on your console terminal appearing directly below your entry. You may see a hard disk error immediately below KERNAL if DFU cannot read the drive's master block on cylinder 0, head 0, sector 1. This means the disk is not formatted.

```
                KERNAL 43: 10/3/86 16:04:02
Accessible Disks:
      Name   Ctlr    Drive     Low    High    Status
  'cdc160mb' 0       0         1      63180   System
  'cdc340mb' 0       1         63181  12732

loading syscalls
loading dfu
loading dfu2
loading dfu3
number of available dbins: 190

Free process pages: 123
DFU Version 43 Thu Jan 15 10:11:31 PST 1987
DFU Modes:
        (0) Exit
        (1) SysFormat.....Format a system disk
        (2) SysCreate.....Create system db on a formatted disk
        (3) SysDestroy.....Destroy the system database
        (4) Merge.....Merge user databases into system
        (5) Validate.....Check correctness of system database
        (6) List.....List  all disks on-line
        (7) Format.....Format a non-system disk
        (8) Unformat.....Remove a disk from the system
        (9) Configure.....Add and delete 'configure' tuples
        (10) Scan.....Remap and mapout sectors
        (11) MarkDb.....Mark recovery status of databases
        (12) Renumber.....Change block numbers of unused disk
        (13) MirrorFormat..Format a disk as a mirror of another
        (14) MirrorCopy....Copy data from one mirror to another
        (15) Compare.....Compare data on mirrored disks
        (16) PromoteDisk...Make a single mirror into a formatted disk
        (17) Rename . . Change the name of a disk
        (18)  FreeZone Free zones on a given disk

Your choice?
```

DFU Modes

SysFormat and SysCreate require that the system database not exist. This prevents the creation of more than one system database. List and Scan modes can be run with or without the system database. All other modes require the existence of the system database.

NOTE

1. The menu appears again only when you enter "?<RETURN>" in response to the prompt, "Your choice?".
2. Modes (13) through (16) appear only when a Mirror DBP board is installed in the Britton Lee database server.
3. The formatting process is basically the same for SysFormat, Format, and MirrorFormat. MirrorFormat has one difference which will be illustrated in the MirrorFormat section.
4. Locate the disk manufacturer's flaw map. You need to use this when a disk is formatted. Flaw maps come with Winchester disk drives and not Removable Packs.

5. You may return to DFU prompt (Your choice?) at any mode prompt by typing "quit<RETURN>".

Each mode is illustrated with the most common examples. Some error messages are also displayed. Select a mode by typing its number. DFU asks you to verify the selected mode. If you answer 'yes',DFU executes the mode and returns to the DFU prompt after completing the operation. DFU is then ready to execute another mode.

EXIT

EXIT will return control to DBP with the Britton Lee database server prompt

```
<BL series> - Filename:
```

When you invoke "0", the IDM system is Reset.

```
Your choice?        0<RETURN>
You chose Exit.
Is this correct?    (y/n) y<RETURN>
Slot 0: 1 meg mem
.
.
.
etc.
```

SYSFORMAT

SysFormat must be used when configuring a Britton Lee database server system. This mode is a combination of the Format and SysCreate modes. Refer to the Format mode for the disk formatting procedures.

```
Your choice?        1<RETURN>
You chose SysFormat mode.
Is this correct?    (y/n) y<RETURN>
(Enter `quit' at any prompt to exit.)
***SysFormat Mode***
```

After completion of the disk formatting, DFU asks whether databases will be created in ASCII or EBCDIC.

Databases can be created in ASCII or EBCDIC representation
(Either 'a' for ASCII or 'e' for EBCDIC)
Character type for database (a/e) a<RETURN>
You chose ASCII.
Is this correct? y/n y<RETURN>

After you have selected the character representation, the SysFormat is completed.

*** SysFormat complete ***
(Type "?" for menu.)
Your choice?

To load the system database, leave DFU with the Exit mode and load the files from diskette or tape. See the *Installation* manual for details on the load procedure.

SYSCREATE

Build a new empty system database on a formatted disk. (SysCreate is similar to the last step of SysFormat.) This mode is disallowed if the system database already exists. This mode can be used to recover from the loss of the system database.
After running SysCreate, the only tuple in the database relation "databases", is that of the system database, and the only tuple in the relation "disk", is that of the system disk. Merge mode of DFU must be run to insert tuples for other databases and disks on-line, and the system software must be reloaded before the Britton Lee database server can become fully operational again.

```
Your choice? 2<RETURN>
You chose SysCreate mode.
Is this correct? (y/n) y<RETURN>
(Enter `quit' at any prompt to exit.)
*** SysCreate Mode ***
Future system disk name? systemtest<RETURN>
This disk has zones numbered 0 through 406.
Zone size is 180 blocks.
Lowest zone number of system database? 0<RETURN>
Allocating system database zones: 0 1 2 3 4 5.
Databases can be created in ASCII or EBCDIC
representation.
(Enter 'a' for ASCII or 'e' for EBCDIC.)
Character type for database? (a/e) a<RETURN>
Default character type is ASCII.
Is this correct? (y/n) y<RETURN>
*** SysCreate complete ***
(Type '?' for menu.)
Your choice?
```

SysFormat creates the system database on the first few zones on the disk. However, SysCreate will ask where you want to create the system database. SysCreate displays an error message if there is not enough space on the disk for the system database or if there is no disk with the given name.

CAUTION

All system and user relations and files in the system database are destroyed when SysDestroy is run. SysCreate does not regenerate user relations or files.

SYSDESTROY

This mode finds and destroys the system database on a given disk. The disk will still be formatted and will still exist in the "disks" relation with corresponding block numbers. SysDestroy can be used to move the System Database to another disk. For example, suppose the disk storage capacity on a Britton Lee database server is being expanded at the expense of other Britton Lee database servers. Several system disks may end up attached to the expanded system. The extra system disks' system Databases may be removed one at a time with SysDestroy.

```
Your choice? 3<RETURN>
You chose SysDestroy mode.
Is this correct? (y/n) y<RETURN>
(Enter `quit' at any prompt to exit.)
*** SysDestroy Mode ***
system disk name? systemtest<RETURN>
system disk  'systemtest  ' found.
freeing system database zones: 0 1 2 3 4 5.
*** SysDestroy complete ***
(Type  '?' for menu.)
Your choice?
```

SysDestroy finds, deallocates, and displays a list of all System Database zones deallocated. The disk is marked as a nonsystem (User) disk.

NOTE

Only 1 system disk may be on-line at a time. Put any other system disks off-line and reboot the system in order to use SysDestroy on an extra system disk.

MERGE

Merge is usually run following SysCreate or SysFormat when there are other disks on-line with user databases. Merge is also necessary when a system has more disk packs (removable media) with databases than disk drives. Each pack must be separately identified.
Merge searches for all on-line disks, including foreign disks. (A disk is on-line when it is ON and READY at the last Britton Lee database server Reset). Each identified disk is compared with a list in the "disks" relation. Those not already in the "disks" relation are merged into the list with their own tuple.

```
Your choice? 4<RETURN>
You chose Merge mode.
Is this correct? (y/n) y<RETURN>
(Enter `quit' at any prompt to exit.)
*** Merge Mode ***
Merging disks...
Disk 'systemtest ' found in 'disks'
Disk 'user1      ' found in 'disks'
Disk 'user2      ' merged in 'disks'
1 disks were merged.
```

In this example, 'systemtest' and 'user1' have tuples in the "disks" relation. Disk 'user2' is new. A tuple is constructed, added to "disks" relation, and "disks" relation status is given.
Each disk in the "disks" relation is searched for existing databases and compared with the tuples in the "databases" relation. Those not already in the "databases" relation are merged as a new tuple.

```
Merging databases...
Database 'system          ' dbid 1 is in 'databases'
relation.
Database 'prospecting     ' dbid 2 is in 'databases'
relation.
Database 'westernregn     ' dbid 3 is in 'databases'
relation.
Database 'easternregn     ' dbid 4 merged in
'databases' relation.
1 database were merged.
*** Merge complete ***
```

Three databases pre-exist in the "databases" relation. (As an example, we will call them: "system", "prospecting", and "westernregn".) The database called "easternregn" is discovered, a tuple constructed, and added to the "databases" relation. Merge also warns of any incomplete databases for which a tuple could not be formed and of any other problems or inconsistencies.

NOTE

Disks from one Britton Lee database server can be merged into another system. However, this can result in overlapping block numbers, duplicate disk and database names, and multiple system disks. See other DFU modes for correction.

VALIDATE

Validate confirms that all on-line user disks and databases have tuples in their respective System Database "disks" and "databases" relations. Validate locates existing databases, attempts to open them, and checks for any databases or disks recorded in the system database that are not on-line.

```
Your choice? 5<RETURN>
You chose Validate mode.
Is this correct? (y/n) y<RETURN>
(Enter `quit' at any prompt to exit.)
*** Validate Mode ***
Validating disks on-line...
Disk 'systemtest    ' found in 'disks' relation.
Disk 'user1         ' found in 'disks' relation.
Disk 'user2         ' found in 'disks' relation.
Virtual disk 'sys1          ' found in 'disks' relation.
Virtual disk 'badtracks     ' found in 'disks' relation.
Virtual disk 'sys2          ' found in 'disks' relation.
All disks on-line are in the 'disks' relation.
Validating 'disks' relation.
Disk 'systemtest    ' is on-line.
Disk 'user1         ' is on-line.
Disk 'user2         ' is on-line.
All disks in 'disks' relation are on-line.
Validating databases on-line...
Database 'system            ' dbid 1 is in 'databases'
relation.
Database 'easternregn       ' dbid 2 is in 'databases'
relation.
Database 'prospecting       ' dbid 3 is in 'databases'
relation.
Database 'westernregn       ' dbid 4 is in 'databases'
relation.
4 databases are in 'databases' relation.
All databases in 'databases' relation are on-line.
*** Validate complete ***
(Type '?' for menu.)
Your Choice?
```

Validate warns of inconsistencies by displaying messages about missing disks, databases, or respective tuples. However, it does not update the system relations, "disks" or "databases". Run Merge to insert any disks or databases that are on-line but not recorded in the system database.

NOTE

Validate should not be confused with CKDB, which tests user as well as system databases. CKDB performs a much more thorough verification of databases than DFU Validate.

LIST

List displays information about disks that are attached to the Britton Lee database server. The name, controller number, drive number, block numbers, and status of each disk on-line are displayed. Any foreign (unformatted) disks are also listed. This mode also displays the name, block numbers, and type of all disks in the "disks" relation, including virtual disks.

```
Your choice? 6<RETURN>
You chose List.
Is this correct? (y/n) y<RETURN>
(Enter `quit' at any prompt to exit.)
*** List mode ***
Disks on-line:
Name          Ctlr       Drive        Low       High          Status
'systemtest    ' 0           0          1      73260       Mirrored system
'user1         ' 0           1      73261     146520
'systemtest    ' 1           0          1      73260       Mirrored system
'user2         ' 1           1     146520     219780
Disks in 'disks' relation:
Name                       Low       High       Type
'badtracks      '        82981      83520          V
'sys1           '        73261      82980          V
'sys2           '        83521     146520          V
'systemtest     '            1      73260          P
'user1          '        73261     146520          P
'user2          '       146521     219780          P
*** List complete ***
(Type '?' for menu.)
Your choice?
```

FORMAT

This mode is used to create a formatted disk. No System Database is created, so the disk would normally be used as a user disk for user databases. SysFormat mode is normally used to create a system disk, but a disk formatted using Format mode can be made into a system disk using the SysCreate option if no System Database exists on the system.

```
Your choice? 7<RETURN>
You chose Format mode.
Is this correct? (y/n) y<RETURN>
(Enter `quit' at any prompt to exit.)
*** Format Mode ***
```

DFU prompts for controller number, drive number, disk name, and whether the controller is SMD or SMDE. (This example assumes the drive sector switches have been set for the correct number of sectors. The Britton Lee database server needs to see a minimum of 2240 bytes).

```
(Enter `quit' at any prompt to exit.)
*** Format Mode ***
Enter information on disk to be formatted:
Disk controller number? (0-3)          0<RETURN>
Disk drive number? (0-15)    1<RETURN>
Name of new disk?  cdc340mb<RETURN>
The controller is either SMD or SMDE.
Formatting SMD drive? (y/n) y <RETURN>
**Formatting SMD drive.**
```

Depending on the controller type, one of the following lists of disks is displayed.

Table 1-4: SMD DISK DRIVES:

	Disk Drive	Capacity	Cylinders	Tracks	Sectors
1	CDC 9710/9762	80 mB	823	5	9
2	9715/9730	160 mB	823	10	9
3	9766	300 mB	823	19	9
4	9715-340	340 mB	711	24	9
5	9775	675 mB	842	40	9
6	Fujitsu m2282	65 mB	823	4	9
7	m2283	130 mB	823	8	9
8	m2284	160 mB	823	10	9
9	CDS t-306	300 mB	823	10	9
10	ams 315	315 mB	845	19	9
11	ams 380	380 mB	845	14	14
12	ams 513	513 mB	845	19	14
13	other drive				

Table 1-5: SMDE DISK DRIVES:

	Disk Drive	Capacity	Cylinders	Tracks	Sectors
1	CDC 9762	80 mB	823	5	8
2	9715-160	160 mB	823	10	8
3	9766	300 mB	823	19	8
4	9715-340	340 mB	711	24	8
5	EMD	368 mB	1217	10	13
6	9715-515	515 mB	711	24	13
7	9775	675 mB	842	40	8
8	9771	825 mB	1024	16	21
9	9772	858 mB	1064	16	21
10	Fujitsu m2333	337 mB	823	10	17
11	2361 (Eagle)	689 mB	842	20	17
12	other drive				

```
Select disk drive from the list or use `other' if not listed.
Selection:
```

If the disk you are formatting is not on the list, or if you wish to use values for cylinders, racks, or sectors that are different than shown for your disk, then use the 'other disks' selection. The program will ask for the values:

```
Selection: 13 <RETURN>
Sectors per track? 9<RETURN>
Tracks per cylinder? 10<RETURN>
180 blocks per zone.
Cylinders per disk? 300<RETURN>
216 blocks per zone.
Disk has 151848 blocks
Low Block: 63181, high block: 215028.
```

DFU asks if the disk should be subdivided into virtual disks. Subdividing the disk makes it look like more than one disk. If the user desires this, Format prompts for the name of each subdivision and the high block number of each subdivision. Subdivision cylinders must equal cylinders per disk minus the remap reserved area which is 1% of available disk space. DFU will prompt if you exceed the highest cylinder number. This example suggests a possible use for virtual disks by assigning subdivision #2 with 6 allegedly bad tracks.

```
Subdivide disk? (y/n) y<RETURN>
You may enter up to 32 subdivisions for the disk.
Cylinders from 0 through 704 are left to be subdivided.
          Name of subdivision?        sub1<RETURN>
   Last cylinder? 250<RETURN>
Cylinders from 251 through 704 are left to be
subdivided.
          Name of subdivision?        sub2<RETURN>
          Last cylinder?     500<RETURN>
Cylinders from 501 through 704 are left to be
subdivided.
          Name of subdivision?        sub3<RETURN>
          Last cylinder?              704<RETURN>
```

```
Disk will be formatted as follows:
Name of disk..... 'systemtest  '
Disk controller number.. 0
Disk drive number..... 0
sectors per track..... 9
tracks per cylinder..... 10
cylinders per disk..... 823
Disk subdivisions:
   Subdivision #1: name = 'sys1                 '  highest
cylinder = 108
   Subdivision #2: name = 'badtracks            '  highest
cylinder = 114
   Subdivision #3: name = 'sys2                 '  highest
cylinder = 815
Is this correct? y/n y<RETURN>
```

If you enter 'n', you will reenter data starting with the drive controller. Or, type "quit<RETURN>" to return to "Your Choice?". If the specified disk is already formatted, Format displays a warning message and asks if the format procedure should continue.

NOTE

If the disk controller is at least Rev. 27, DFU asks whether special formatting patterns are required. This allows more thorough testing of the disk's magnetic properties and helps detect defective sectors. The user may enter up to four hexadecimal patterns of four digits each.

```
Do you wish special formatting? y/n y<RETURN>
Enter up to 4 patterns consisting of four hex digits
Pattern: d9b3<RETURN>
Another pattern? y<RETURN>
Pattern: 366c<RETURN>
Another pattern? n<RETURN>
2 patterns entered
Pattern 1 = D9B3
Pattern 2 = 366C
Is this correct? (y/n): y<RETURN>
Disk divided into 407 zones.
```

The formatting process divides each disk into zones and blocks. In this example, 407 zones will be reported in the master block. A disk's sector contains a block of 2048 data and 192 overhead bytes. The exact number of blocks per zone is determined by the format program, optimized for the kind of disk being formatted. Note that DFU's own formatting pattern (DB6C) follows any requested special patterns.

DFU displays the approximate time to complete formatting. The time is based on 3600rpm, and is incorrect for other speeds. The estimated time does not include time for remapping of any bad blocks. About every fifteen seconds, DFU displays the number of cylinders left to be formatted.

```
Approximate format time (at 3600 rpm): 17 minutes, 45
seconds with 2 special patterns.  814 cylinders left to be
formatted.
```

Any corrupted block (sector) found during format is locked out. REMAP area corrupted blocks are mapped out. WORKING area corrupted blocks are remapped to the REMAP area. (Two examples are given for clarity). Mark all errors on the disk manufacturer's flaw map.

```
Disk hard err code xC000
cmd 14, ctlr 0, drive 0, cyl 818, head 5, sec 4
    errcode 101, errstat 0, phyaddr 0
cyl 818, head 5, sec 4 mapped out.

22 cylinders left to be formatted.
Do you wish to mark any sectors as defective? y/n y<RETURN>
```

When DFU has finished formatting the surfaces, it politely reminds you now is the time to use the disk's flaw map. A flaw map has columns to identify cylinders, heads (or tracks), and byte position from index. The map may also show the number of bit errors at that byte position. If the number of bits is reported, then only remap flaws with 3 or more bits not already remapped by DFU. Calculate the flawed sector with this formula: sector # = byte position/2240

(The constant is Britton Lee database server 2K sector with 192 byte overhead). Discard any fraction. Numbering starts with sector 0. If this disk's flaw map lists cylinder 27, head 4, and 12512 bytes, then 12512/2240 = 5.587. Discard 0.587 and use sector 5.

NOTE

Century Data systems ams-513 drives use 2270 bytes per sector.

```
Enter cylinder?              27<RETURN>
Enter head?    4<RETURN>
Enter sector?  5<RETURN>

Cyl 27, head 4, sector 5, remapped to cyl 814,
head 0, sector 1, on Ctlr 0, Drive 0.
Another? y/n n<RETURN>

2 blocks were remapped
1 block was mapped out
```

The last statement includes the two previous mapping examples.

DFU checks the cylinder and head selection on the drive. If any of the cylinder or head selection lines are shorted or open, Format displays a warning message. If all is still well, DFU builds the master block of the disk and writes it on the second good block of the disk.

UNFORMAT

This mode deallocates blocks on a disk and logically removes the disk from the system by marking it as a foreign disk. The disk's tuple is removed from the "disks" relation. Unformat scans the disk to be unformatted looking for allocated block numbers, which are then freed. If the specified disk is a system disk, run SysDestroy followed by Unformat.

```
Your choice? 8<RETURN>
You chose Unformat mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** Unformat Mode ***
Name of disk to be unformatted? systemtest<RETURN>
WARNING: system database does not exist.
    Unformatting will only mark disk foreign.
Continue unformatting? (y/n)      y<RETURN>
WARNING: zone 4 belongs to dbid 2.
Unformatting will destroy all data on this disk.
Continue unformatting? (y/n)      y<RETURN>
Marking this disk 'foreign'.
Deleting 'disks' tuple(s).  *
** Unformat complete ***
(Type '?' for menu.)
Your choice?
```

If the disk is not on-line, but has an entry in the "disks" relation, Unformat asks if the process should continue. If you respond 'y', Unformat deletes the disk's tuple in the "disks" relation and returns to the main menu.
If the disk does not have an entry in the "disks" relation, but is on-line, Unformat asks if the process should continue, and if it does, it will still Unformat the disk.
Unformat checks all zones of a disk that is to be unformatted for any user databases and displays a list of any that are found. It then asks if the process should continue. If you respond 'y', any databases that have zones on the disk will be marked as 'unrecoverable' and you should destroy them later.
After unformatting, the disk will be marked as 'foreign' to the system.

CONFIGURE

Configure is used to insert, change, and delete "configure" tuples. Any value, even a nonsense value, may be inserted in a "configure" relation tuple field. A Britton Lee database server doesn't recognize a nonsense value until the tuple is loaded. False information could render a Britton Lee database server completely unusable.
Configure mode lets you add new tuples, change or delete existing tuples. Configure operational detail is found in the *Installation Manual* and the *System Administrator's Manual*. In the following example, Serial port #6 is turned off.

```
Your choice? 9<RETURN>
You chose Configure Mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** Configure Mode ***
Configure Relation.
type .... D   number ....... 0 value . 0
type .... M   number ....... 0 value . 1
type .... R   number ....... 0 value . 35
Configure Mode commands
    (0) Return..... Return to Main Program
    (1) Add........ Add a tuple
    (2) Delete..... Delete a tuple
Your choice? 1<RETURN>
You chose Add.
Is this correct? (y/n) y<RETURN>
Enter 'configure' tuple values:
    Type? S<RETURN>
    Number?   6<RETURN>
    Value?    11<RETURN>
You entered this 'configure' tuple
type .... S   number ....... 6 value . 11
Is this correct? (y/n) y<RETURN>
Tuple inserted.
Configure Relation.
type .... D   number ....... 0 value . 0
type .... M   number ....... 0 value . 1
type .... R   number ....... 0 value . 35
type .... S   number ....... 6 value . 11
Configure Mode commands
    (0) Return..... Return to Main Program
    (1) Add........ Add a tuple
    (2) Delete..... Delete a tuple
Your choice? 0<RETURN>
You chose return.
Is this correct? (y/n) y<RETURN>
*** Configure complete ***
(Type '?' for menu.)
Your choice?
```

SCAN

CAUTION

All databases must be SAFE prior to running scan mode. If scan is used on a Britton Lee database server system with unsafe databases (no recovery since the last time the system was brought down), data may be lost if blocks are remapped or mapped out by scan. The system database is never "already safe", but it must have had recovery run on it anyway.

If the system Database is being recreated, you should run MERGE and bring the system software up to SAFE prior to running scan to ensure that all databases are safe. This means you must EXIT from DFU.

Scan's job is to scan (read) a disk looking for bad sectors. The disk is divided into zones when first formatted. The first good sector of each zone is the blockalloc page. Blockalloc's function is to keep the track of free, allocated, and defective sectors within the zone. Three types of disk read problems on the Britton Lee database server are identified.
disk soft error A008, error code 0
disk soft error B008, error code 0
disk hard error E00X, error code 100

When a soft error occurs consistently on a particular block of a disk, you should remap the bad block with scan as soon as the Britton Lee database server can be dropped to MAINT Mode. This avoids reformatting the disk and reloading the databases that were on the disk. Disk hard errors cannot be remapped if the errored sector is allocated to a database.

SOFT ERRORS. Recovery from a failed sector, called a soft error, is shown with an "error code 0". The "B" category soft error may have characters other than "B00". However, the trailing "8" is always true.

A008 is a return code from the Disk Controller I/O register. The code means 1 of 8 data subsectors of the stated sector had up to 5 sequential bit errors when read. The command was retried and the sector was read correctly on the second attempt. The digit "8" means that all 8, 256-byte subsectors were processed correctly.

B008 return code is similar to the A008 return code . A soft read error occurred. Read was re-executed and was still soft on the second read. Disk Controller ECC (error correction code) circuitry was enabled, allowing up to 60 additional read retries before the subsector was fixed. The digit "8" means that all 8, 256-byte subsectors were eventually processed correctly.

HARD ERRORS. More than 5 bits in error constitutes a hard read error: unrecoverable.

E00X return code is a sample only. The first three characters are bit sensitive, (see the *Console Message Summary* for a complete list of possible meanings). "X" represents a number between 0 and 8 to show how many subsectors were processed before the bad subsector was encountered. The message means that a subsector within the sector read had more than 5 bits in error. The Disk Controller's ECC circuitry could not correct the data with any certainty. Since the sector may not be recovered without **GREAT** difficulty, usually it is unrecoverable.
Scan was devised to run in three modes: manual mapping, automatic scanning without mapping, and automatic scanning with mapping. Both automatic and manual scan use the same remapping routines that SysFormat, Format, and MirrorFormat modes use.

Manual Mapping mode. The Manual mode of scan is used to map out a soft allocated sector without scanning a complete disk. Typically, this mode is used to quickly map out a recurring soft read error. Unlike the other scan modes, Manual allows an operator to remap hard or soft failed sectors IF the sector IS MARKED free IN blockalloc. An allocated hard error sector cannot be remapped.

```
Your choice? 10<RETURN>
You chose Scan mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** Scan Mode ***
Name of disk to be scanned?        systemtest<RETURN>
1676 free out of 1728 in remap zone.
Manually map out sectors? (y/n) y<RETURN>
    cylinder? . 5<RETURN>
    head? 16<RETURN>
    sector?     7<RETURN>
Cyl 5, head 16, sector 7, remapped to cyl 814,
head 0, sector 1, Ctlr 0, drive 0.
Another? (y/n) y<RETURN>
    cylinder?   5<RETURN>
    head? 16<RETURN>
    sector?     7<RETURN>
Remapping a remapped block: 1231.
cyl 5, head 16, sec 7, remapped to
cyl 814, head 0, sec 2, on ctlr 0, drive 1
Another? (y/n) n<RETURN>
*** Scan complete ***
```

In this example, the Manual Mode is used to remap a defective sector. In a mistake, the same sector is remapped again. Scan complains but remaps to another sector. Remap sector 814,0,1 will not be available because of the double remap.

Automatic mode with no remap capability This mode also scans a specified disk. However, it does not remap sectors. It just makes reports to the console port and continues. This mode was designed in case the drive being scanned had broken hardware. In this mode, scan read without remapping and noted two soft errors. This mode is particularly useful if there is reason to believe the disk drive is defective. It is conceivable that DFU might attempt to remap or map out every allocated sector that it reads, unable to discern that something other than bad media is the cause.

```
*** Scan Mode ***
Name of disk to be scanned?        systemtest<RETURN>
1676 free out of 1728 in remap zone.
Manually map out sectors? (y/n) n<RETURN>
Report errors without remapping? (y/n) y<RETURN>
Number of scans to be performed? 1<RETURN>
Scanning zone 0.
.
.
Disk soft err code xA008
cmd 3, ctlr 0, drive 1, cyl 518, head 4, sec 7
    errcode 0, errstat 1, phyaddr 14C800

WARNING: soft read error on block 85796.
    Not remapping the block.

Scanning zone 400.

Disk soft err code xB008
```

```
        cmd 3, ctlr 0, drive 1, cyl 397, head 1, sec 1
            errcode 0, errstat 1, phyaddr 18A800

WARNING: soft read error on block 111899.
   Not remapping the block.
Scanning zone 520.
0 blocks were reported bad.
2 soft read errors.
*** Scan complete ***
```

In this example, scan read without remapping and noted two soft errors.

Automatic mode with remap capability This mode scans (reads) each sector on a specified disk. When it encounters a soft read error, it inspects that zone's blockalloc block to see if the sector is allocated, free or bad. If it is allocated, scan will retry the read. If it is still soft (B008 category above), then scan will move the data to the first free sector in the reserved area. The original sector is marked defective in its zone's blockalloc Block. If the retry read was A008 category (above) or blockalloc block says that the sector is free, then scan continues without making changes. If a permanent read error is encountered on an allocated sector (E00X category above), scan can do nothing with it since the contents of the sector are unknown.

```
*** Scan Mode ***
Name of disk to be scanned?        systemtest<RETURN>
1676 free out of 1728 in remap zone.
Manually map out sectors? (y/n) n<RETURN>
Report errors without remapping? (y/n) n<RETURN>
Number of scans to be performed? 1<RETURN>
Scanning zone 0.
.
.
Disk soft err code xB008
cmd 3, ctlr 0, drive 1, cyl 397, head 1, sec 1
    errcode 0, errstat 1, phyaddr 18A800
cyl 518, head 1, sec 1 remapped to
cyl 814, head 0, sec 2.
1 block was remapped
0 blocks were mapped out.
*** Scan complete ***
(Type '?' for menu.)
Your choice?
```

MARKDB

This mode lets the user mark user databases as recoverable or unrecoverable. It is useful if a user database has been damaged in a way that causes the recover program to fail. The System Database cannot be marked unrecoverable. A database marked unrecoverable cannot be opened but can be "destroyed". Use CKDB to analyze database damage. Results of this mode are seen when recovery operates after the system files are loaded. The specified database is listed as "unrecoverable". All other databases are listed as "recovered" or "already safe".

```
Your choice? (y/n) 11<RETURN>
You chose MarkDb mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** MarkDb Mode ***
Databases:
    'system      ' is marked recoverable.
    'easternregn             ' is marked recoverable.
    'prospecting' is marked recoverable.
    'westernregn             ' is marked recoverable.
MarkDb mode commands:
    (0) Return to Main Program
    (1) Mark a database unrecoverable
    (2) Mark a database recoverable
Your choice? (0-2): 1<RETURN>
You chose the Mark Unrecoverable command.
Is this correct? (y/n): y<RETURN>
Name of database to be marked? easternregn<RETURN>
easternregn marked unrecoverable.
MarkDb mode commands:
    (0) RETURN to Main Program
    (1) Mark a database unrecoverable
    (2) Mark a database recoverable
Your choice? (0-2): 0<RETURN>
You chose the Return to Main Program command.
Is this correct? (y/n): y<RETURN>
*** MarkDb complete ***
(Type '?' for menu.)
Your choice?
```

RENUMBER

This mode allows changing the block numbers of a disk, provided there is no information on the disk. This is useful if a disk is removed and the block numbers used by it must be reused. A formatted disk of the same size with no information on it can be renumbered to use the same block numbers as the old one. This changes the block numbers on the master block of the disk, the 'disks' tuple(s) and the page numbers of the blockallocs on each zone. In most cases, Renumber saves reformatting because of overlapping blocks.

```
Your choice? 12<RETURN>
You chose Renumber mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** Renumber Mode ***
Name of disk to be renumbered? local<RETURN>
New low block number?       219781<RETURN>
*** Renumber complete ***
(Type '?' for menu.)
Your choice?
```

Mirror Operation

Mirror operation requires a Mirror DBP and disk controllers with Rev 26 or higher firmware. Modes 13 through 16 Format, Copy, Compare, and Promote (make single) disks. The mirror feature uses disk pairs that are exact copies. During operation, DBP and the DC's assure correct WRITE and READ operations. Upon failure of one disk, DBP informs the console port. If a Mirrored disk is corrupted, it is marked foreign and operation automatically continues with a "promoted" single disk.

MIRRORFORMAT

A Britton Lee database server can mirror if this is reported on the Console Terminal at system Reset time:

```
    Mirrored  DBP rev 30 (6MHz)
or  Mirrored  DBP rev 31 (10MHz)
    Slot 0:1Meg mem
    Slot 1:1Meg mem
    Slot 2:t&c rev 3
    Slot 5:serial chan rev 38
    Slot 6:parallel chan rev 26
    Slot 7:block mux rev 0
    Slot 8:ethernet rev 0
    Slot 9:tpc rev 107
    Slot 11:da rev 37
    Slot 12:dc rev 28
    Slot 13:dc rev 28
    Slot 14:dbp rev 30M (6MHz)
or  Slot 14:dbp rev 31M (10MHz)
```

```
<BL series> - Filename:
```

SysFormat and Format are similar to MirrorFormat. In MirrorFormat, DFU does not ask for cylinder, head, and sector because DFU expects the same type disk to be used. Different sized disks may be formatted so long as the SMALLEST disk is SysFormatted or Formatted and the LARGER is MirrorFormatted. The larger disk must have an equal or larger number of tracks per cylinder, cylinders per disk, and sectors per track. The larger disk will have wasted space as the total number of blocks are based on the smaller disk.
However the initial disk is formatted, the MirrorFormatting must be the same. (Disk subdivision is omitted below). Recommended: separate disk controllers.

```
Your choice? 13<RETURN>
You chose MirrorFormat mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** MirrorFormat Mode ***
Enter information on disk to be mirror formatted:
Disk controller number? (0-3) 1<RETURN>
Disk drive number? (0-15) 0<RETURN>
Enter information on existing disk:
Name of existing disk? systemtest<RETURN>
Disk will be formatted as follows:
Name of disk..... 'systemtest  '
Disk controller number.. 1
Disk drive number..... 0
sectors per track..... 9
tracks per cylinder..... 10
cylinders per disk..... 823
```

```
Is this correct? y/n y<RETURN>
```

The remainder of MirrorFormat is the same as SysFormat and Format.

NOTE

MirrorCopy always follows MirrorFormat.

MIRRORCOPY

This mode identifies both disks and copies files from the first disk onto the second or "mirrored" disk. This is file copying, NOT sector by sector copying. Each disk has its own physical characteristics for remapping and is transparent to the user. Only allocated sectors are copied.
This mode assumes the original disk has been loaded with its system or user files before attempting to MirrorCopy.

```
Your choice? 14<RETURN>
You chose MirrorCopy mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** MirrorCopy Mode ***
Name of disk to be copied? systemtest<RETURN>
Copying 'systemtest  ' from ctlr 0 drive 0 to ctlr 1, drive 0.
Copying zone 0.
.
.
.
Copying zone 400.
*** MirrorCopy complete ***
(Type '?' for menu.)
Your choice?
```

COMPARE

This mode compares the mirrored disks, checking for inconsistencies. Compare should be selected following MirrorCopy when disks are originally mirrored and any time a disk is recovered from corruption with MirrorCopy.

NOTE

This mode may also be used to compare other duplicate disks, even with different names.

```
Your choice? 15<RETURN>
You chose Compare mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** Compare Mode ***
Name of first disk? systemtest<RETURN>
Controller number? (0-3) 0<RETURN>
Drive number? (0-15) 0<RETURN>
Name of second disk? systemtest<RETURN>
Controller number? (0-3) 1<RETURN>
Drive number? (0-15) 0<RETURN>
Comparing zone 0.
.
.
.
Comparing zone 400.
Disks are identical.
*** Compare complete ***
```

PROMOTE DISK

Promote disk lets you keep a single user or system disk operating after one mirrored disk has failed. When a mirrored disk fails, DBP reports the problem to the console port.

```
Your choice? 16<RETURN>
You chose Promote mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** Promote Mode ***
Name of disk to be promoted?    user1<RETURN>
*** Promote complete ***
(Type '?' for menu.)
Your choice?
```

RENAME

This mode allows a physical disk and any virtual disk to be given new names.

```
Your choice? 17<RETUTN>
You chose Rename mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** Rename Mode ***
Name of disk to be renamed? cdc340mb<RETURN>
Physical disk 'cdc340mb'.
Change name? (y/n) y<RETURN.
New name: 1cdc340mb<RETURN>
Virtual disk 'sub1'.
Change name? (y/n) y<RETURN.
New name: sub1x<RETURN>
Virtual disk 'sub2
Change name? (y/n) n<RETURN>
Virtual disk 'sub3'.
Change name? (y/n) n<RETURN>
```

At this point, the program will display the names of the physical and virtual disks including any changes that you have made. If you respond that the names are correct, they will become the current names.

```
Physical disk '1cdc340mb'.
Virtual disk 'sub1x'.
Virtual disk 'sub2'.
Virtual disk 'sub3',
Is this correct? (y/n) y<RETURN>
*** Rename Complete ***
(Type '?' for menu.)
Your choice?
```

FREEZONES

This mode is used to deallocate zones on a disk that are assigned to a database. It is used mainly to take care of zones on a disk that remain when a disk is moved between system, or when a disk was not on-line when a database is destroyed. This mode cannot be used if the database has "disk_usage" relation entries that point to the disk. To deallocate these zones, the database should be destroyed.

```
Your choice? 18 <RETURN>
You chose FreeZone mode.
Is this correct? (y/n) y<RETURN>
(Enter 'quit' at any prompt to exit.)
*** FreeZones mode ***
Dbid:       9<RETURN>
Disk name: cdc340mb<RETURN>

WARNING:    All zones on disk 'cdc340mb' with dbid 9 will be
freed.
            The database will be destroyed on that disk.
```

Dfu Usage Examples

Suppose a user requires five databases, V, W, X, Y, and Z, on three disks, D1, D2, and D3. To create this system, the user must perform the following steps:

- In MAINT Mode, format the system disk (D1) using DFU SysFormat.
- Leave DFU and MAINT Mode for RUN Mode to create user databases V, W, X, Y, and Z on the disks using the IDL or SQL parser through an I/O Channel's Host Computer. There is usually enough space on the system disk for user databases in addition to the System database.
- Re-enter MAINT Mode to use DFU Validate to ensure the System Database is consistent with the user databases and disks on-line. This is optional, but should be done when the system is first created or whenever it is rebuilt.

Suppose the user has the following:
System database on disk D1
user database V on disk D1
user database W on disks D1, D2, and D3
user database X on disk D2
user database Y on disks D2 and D3
user database Z on disk D3

You can use several modes of DFU in combination to recover from almost any disk defect or crash. DFU has other useful features for maintenance of the disks on-line and for adding new disks to the system. Given the above configuration, some possible scenarios requiring DFU are explained below.

EXAMPLE 1

Disk D2 is lost or damaged beyond use, but the system disk D1 and System database are still intact. To recover, the user must:

- In RUN Mode via a host, destroy any databases that were partially or completely on D2 (W, X, and Y). This removes any traces of the missing databases from the System Database.
- In MAINT Mode with DFU, Unformat D2 to free the block numbers that were formerly allocated to D2. This is required only if too few unallocated block numbers remain for the new disk.
- In MAINT Mode with DFU, format a new disk, D2" with Format.
- In RUN Mode via a host, create and load databases W, X, and Y, and roll forward as necessary. User databases need not be placed on D2" if there is enough space elsewhere.

EXAMPLE 2

The system disk D1 is lost or damaged beyond use. The System database and user databases V and W have been lost along with the disk. To recover:

- In MAINT Mode with DFU, use SysFormat to create a new system disk.
- In MAINT Mode with DFU, Merge D2 and D3 into the new System Database. This adds information about the User Databases on D2 and D3 to the System Database.
- In MAINT Mode, leave DFU and load the System Database from the source media on a new disk, D1".
- In RUN Mode through a Host, load databases V and W and roll forward as necessary. It may be necessary to destroy V and W before loading if any zones of D2 and D3 are allocated to V or W.

Alternatively:

- Run SysCreate on one of the remaining disks on-line. This may be done if there is enough space on that disk for databases V and W as well as the System database.
- Merge D2 and D3 into the system.
- Load databases V and W and rollforward as required.

EXAMPLE 3

The user wants to replace the system disk D1 with a new disk, D4. A new disk may be required if D1 is of insufficient size, has too many defective blocks, or has been damaged. The user must:

- In RUN Mode through a Host, dump databases V and W. Destroy databases V and W.
- In MAINT Mode with DFU, run SysDestroy on D1. If necessary, UnFormat D1 and remove it.
- Install D4 and in MAINT Mode, run DFU SysFormat on it.
- In MAINT Mode, leave DFU and load the System Database from the source media.
- In RUN Mode through a Host, create and load V and W.
- In MAINT Mode with DFU, Merge D2 and D3 into the System Database.

EXAMPLE 4

Suppose you have created a new System database and there are more disk packs containing user databases than there are drives connected to the Britton Lee database server. You must Merge all the databases into the System database. While not all user databases can be on-line at once, the System database contains information on all databases. Should you attempt to access a database that is not currently on-line, the Britton Lee database server returns an error message to the host program.
Given two drives, D0 and D1, and three disk packs P1, P2, and P3, complete the following:

- Format the system disk on drive D0.
- Insert pack P1 in drive D1 and run Merge on that drive.
- Remove P1 and repeat with packs P2 and P3, leaving P3 in drive D1.
- Run Validate to determine which databases are offline along with packs P1 and P2.

EXAMPLE 5

Disks from Britton Lee database server A must be moved to Britton Lee database server B to increase the storage available to Britton Lee database server B. After connecting them to a disk controller, the disks are immediately added to the list kept by DFU of on-line disks.
The correct method to transfer databases between Britton Lee database servers is to:

- dump the databases from Britton Lee database server A to tape
- load the databases from tape to Britton Lee database server B.

The proper method of transferring disks between Britton Lee database servers is:

- Destroy databases on the disks being moved with **drop database** (Refer to the *SQL Reference Manual* for details on the **drop database** command). If a system disk is being moved, DFU SysDestroy should be used before attempting the move.
- Move the disks to Britton Lee database server B.
- Connect the drives to Britton Lee database server B and use DFU List mode to ensure the drives are connected properly.
- Use DFU Renumber to make the new blocks contiguous.
- Use DFU to Merge the newly added disks names and block numbers to the System Database of Britton Lee database server B.
- After a user disk is moved, run UnFormat on Britton Lee database server A to remove the old disk blocks and name from the System database.

Several problems may result from transferring disks if the above procedure is not followed: overlapping block numbers, multiple system disks, and incomplete databases remaining on the transferred disks. The List Command displays the status of each disk, showing any problems.
Solve the overlapping block number problem with Renumber on a disk whose block numbers overlap. Duplicate disk name and incomplete database problems are solved with UnFormat. Follow with Format. Merge is useless if the block numbers overlap.
Format is useless in the multiple system disk problem. It disallows formatting a system disk. SysDestroy must be used to "unsystem" one or more of the system disks after putting any other system disks off-line and rebooting. The remaining system disk will become the new system disk for Britton Lee database server B.

If there are multiple system disks with overlapping block numbers, then after using SysDestroy, Renumber should then be used to assign distinct block numbers to the overlapping disks.

INDEX

E

F

H

I

L

M

O

P

Q

R

S

T

U

Z