HP 3000 Computer Systems

HEWLETT PACKARD

IMAGE/3000
Student Workbook

Course No. 35053A
Part No. 35053–90001

OCT 1982

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HP Computer Systems Training Course

IMAGE/3000

Student Workbook

HEWLETT
PACKARD

19310 Pruneridge Ave., Cupertino, California 95014

Part No. 35053-90001 Printed in U.S.A. 8/82

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

IMAGE/3000

A

DATA BASE MANAGEMENT SYSTEM

notes:

references:

TOPICS

* COURSE DESCRIPTION
* COURSE OUTLINE
* COURSE SCHEDULE

notes:

references:

COURSE DESCRIPTION

COURSE NUMBER: 35053A

LENGTH: 5 days

PREREQUISITES: A Programmer's Introduction and experience in programming on HP3000.

INTENDED AUDIENCE: Database administrators/designers, database application programmers and systems analysts.

OBJECTIVES: Upon successful completion of this course the student should be able to:

1. Understand data base concepts.
2. Define and create IMAGE data bases.
3. Access IMAGE data bases from application programs utilizing IMAGE procedures.
4. Access IMAGE data bases using QUERY/3000 as a program development tool.
5. Understand multiple user considerations and implement locking to coordinate concurrent access.
6. Specify security provisions at various levels within the data base.
7. Implement data base logging and recovery using the IMAGE utilities.
8. Understand IMAGE data base design and performance considerations.
9. Implement remote data base access.
10. Understand how RAPID/3000 relates to effective data management on HP3000 system.

STUDENT MATERIALS:

32215-90003	IMAGE/3000 Reference Manual
35053-90001	IMAGE/3000 Student Workbook
30000-90049	MPE Pocket Guide

COURSE OUTLINE

MONDAY

I. Introduction

II. Presentation Strategy: Case study

III. Data Management

IV. IMAGE Data Base Management overview

LAB 1: Worksession

V. IMAGE and the Case Study

LAB 2: Students interact with the final version of the case study data base as users.

VI. IMAGE implementation: An overview

VII. QUERY/3000 as a program development tool

LAB 3: QUERY commands

TUESDAY

VIII. Data base creation and schema analysis

LAB 4: Design and create a simple data base

IX. Data base access:

(a) Procedures: OPEN, PUT, INFO, EXPLAIN, ERROR, CLOSE

LAB 5: Use calls; check status

(b) Procedures: FIND, GET; access modes

LAB 6: Use calls

WEDNESDAY

(c) Procedures: UPDATE, DELETE

LAB 7: Use calls

(d) Multiple user considerations: Open modes, locking DBLOCK, DBUNLOCK

LAB 8: Use calls

(e) Security; types of users

LAB 9: Use set and item security

THURSDAY

X. (a) Utilities and Restructuring: DBUTIL, DBSTORE, DBRESTOR, DBUNLOAD, DBLOAD.

LAB 10: Worksession; instructor demo

(b) Transaction logging and recovery (logging cycle): DBBEGIN, DBEND, DBMEMO; DBRECOV

LAB 11: Worksession

XI. IMAGE internals

XII. Performance considerations

LAB 12: Worksession

FRIDAY

XIII. Design considerations

XIV. Miscalleneous:

(a) Remote data base access

(b) IMAGE and RAPID/3000

(c) [illegible]

(d) [illegible]

COURSE SCHEDULE

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
Introduction Case study Data mngmt IMAGE DBMS LAB 1	DB creation & schema LAB 4	LAB 6 (cont'd) UPDATE, DELETE LAB 7	Utilities & restructuring LAB 10 Logging & recovery LAB 11	Design Misc.: RDBA RAPID
IMAGE & case: LAB 2 IMAGE imp. QUERY LAB 3	DB access: OPEN, PUT, etc. LAB 5 FIND, GET, access modes LAB 6	Multiple user considerations: open modes, locking LAB 8 Security LAB 9	Internals Performance LAB 12	
mostly lecture	½ lab	½ lab	2½ lab	

notes:

references:

IMAGE/3000

PRESENTATION STRATEGY

notes:

references:

IMAGE/3000
PRESENTATION STRATEGY

To relate learning IMAGE to solving a typical business problem with a

CASE STUDY

notes:

references:

THE CASE STUDY

* A TYPICAL BUSINESS PROBLEM *

WHO?	A successful real estate brokerage...WONDER REALTY
WHAT?	The need to match properties with buyer desires accurately and quickly
WHEN?	Today
WHERE?	Anywhere — Planet Earth
WHY?	Increase sales, increase profits, increase customer satisfaction
HOW?	Computers

notes:

references:

WONDER REALTY'S OBJECTIVES

* Minimize the sales cycle by quickly and accurately presenting buyers with properties that match their stated wants and needs

* Implement the ability to list properties outside the local area

* Expand the business to offices in areas serving major corporations' employee relocation needs

notes:

references:

WONDER REALTY'S PROPOSED SOLUTION

Purchase a computer system for storage and retrieval of properties information

notes:

references:

WONDER REALTY

* Residential and Commercial properties
* ~$25 million gross revenues
* Desire to automate
* Remote offices planned
* Relocation service planned
* HP 3000 "chosen system"

notes: Commercial - Residential properties

references:

WONDER REALTY

wants a computer solution to get immediate information.
So...

WHERE TO START?

hardware has been chosen, [illegible]
[illegible]
— 'Data Management' as our first topic

notes:

references:

DATA

MANAGEMENT

is all about storing, retrieving, adding, updating, deleting data correctly ... as is the case with paper files.

notes:

references:

TOPICS

* INFORMATION RETRIEVAL
  - Accessing files
  - Data Base access methods

* WHAT IS DATA BASE?

* WHY DATA BASE?

notes:

references:

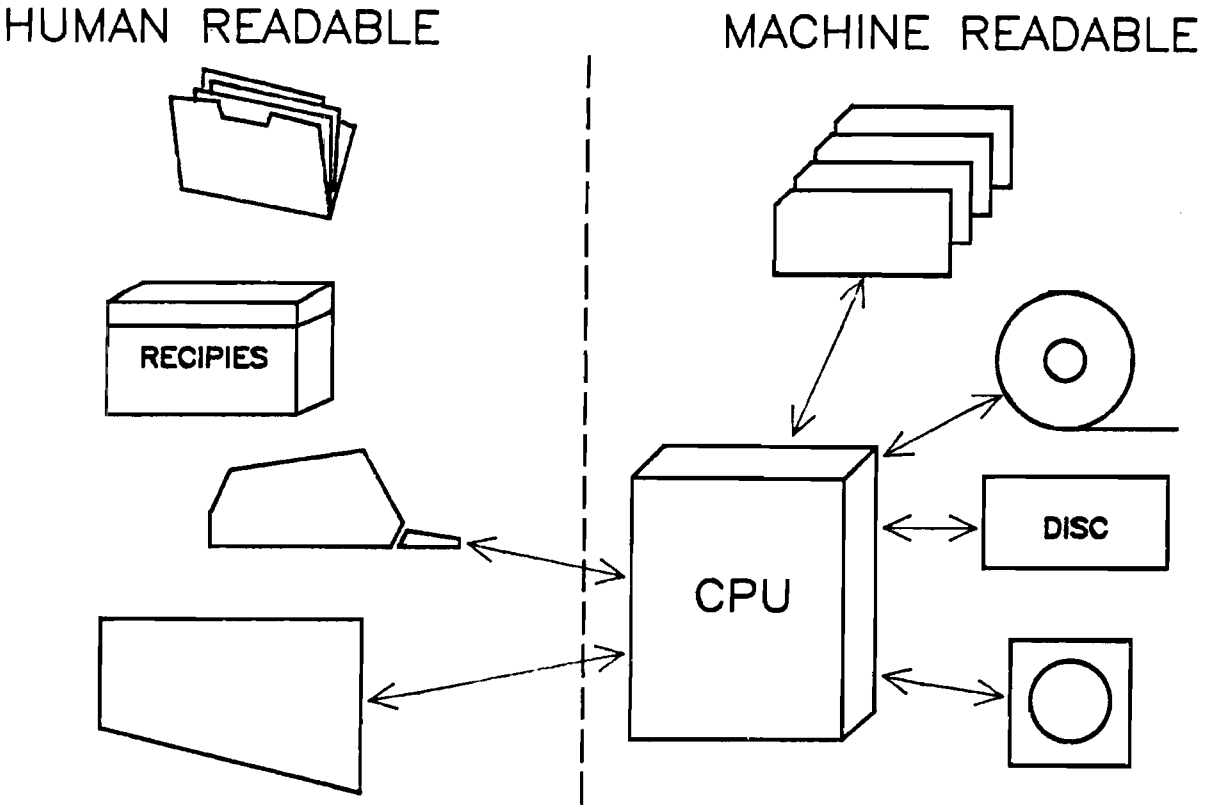
INFORMATION RETRIEVAL

* Accessing Files
* Data Base Access Methods

notes:

references:

DATA MANAGEMENT – INFORMATION



notes:

Some forms of data storage —

Data items are stored in records with other associated data (part and) such that one record consists of many items of data that individually have little meaning. Many associated data items give meaning to each other — concept of information.

Manila file folder	Punched cards
Card file (recipes)	Magnetic tape
CRT terminal	Magnetic disc
Computer printout	Diskette (floppy disc)

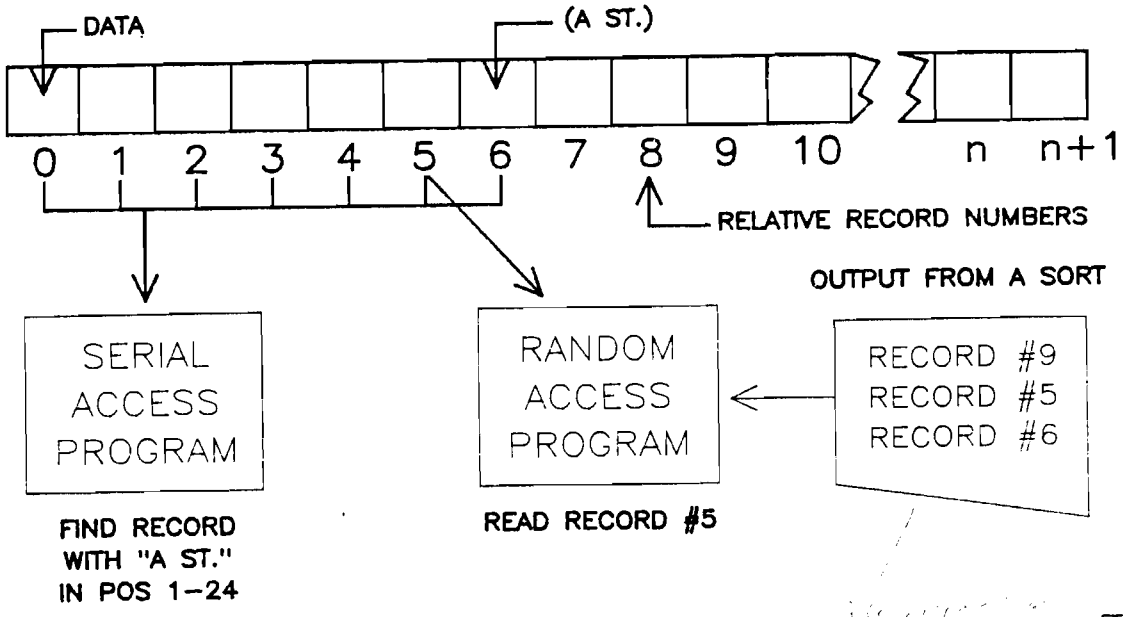
- many forms of information storage, in 'files' for ease of access

- [illegible] so that it can be retrieved quickly —

references:

INFORMATION RETRIEVAL
ACCESSING FILES

* SERIAL – Record by record in order of occurrence
* RANDOM (DIRECTED) – Selection of a specific record



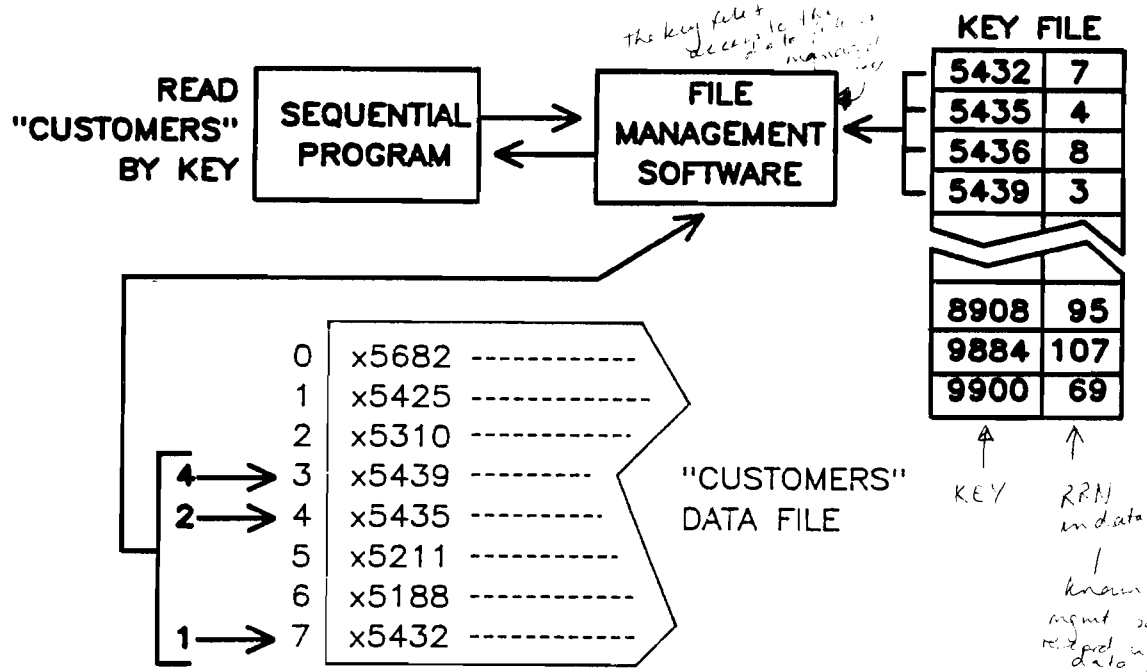
notes:

- most computer systems (even micros) offer both serial and random disc access via their "file systems".

references:

ACCESSING FILES

* SEQUENTIAL (BY KEY) — Records read in ascending sequence based on "key" field value.



...BEYOND SEQUENTIAL...

notes:

- sequential is a combination of serial and direct access
- sequential is also referred to as 'Sequential by Key', 'Keyed', or 'Indexed Sequential' access.
- Examples of sequential file management software:
  KSAM - Keyed Sequential Access Method (HP3000)
  ISAM - Indexed Sequential Access Method (IBM)
  VSAM - Virtual Sequential Access Method (IBM)

references:

DATA BASE ACCESS METHODS

* SERIAL

* DIRECTED

* CALCULATED

* CHAINED

notes:

- these four access types are implemented in IMAGE

(p 4-10 INFO ref manual

references:

DATA BASE ACCESS - IMAGE implementation

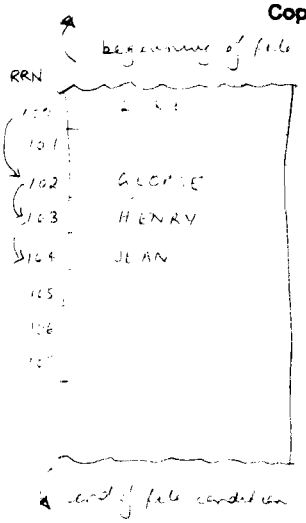
* SERIAL
  - First to Last - FORWARDS
  - Last to First - BACKWARDS - sometimes faster when application needs to access most current data or latest additions - flexibility
* DIRECTED
  - Synonomous with Random
  - By Relative Record Number

Flexibility - been left in the access when a program has already read records via other access methods and needs to access the same record again - least used type of access

HEWLETT PACKARD

notes:

SERIAL -

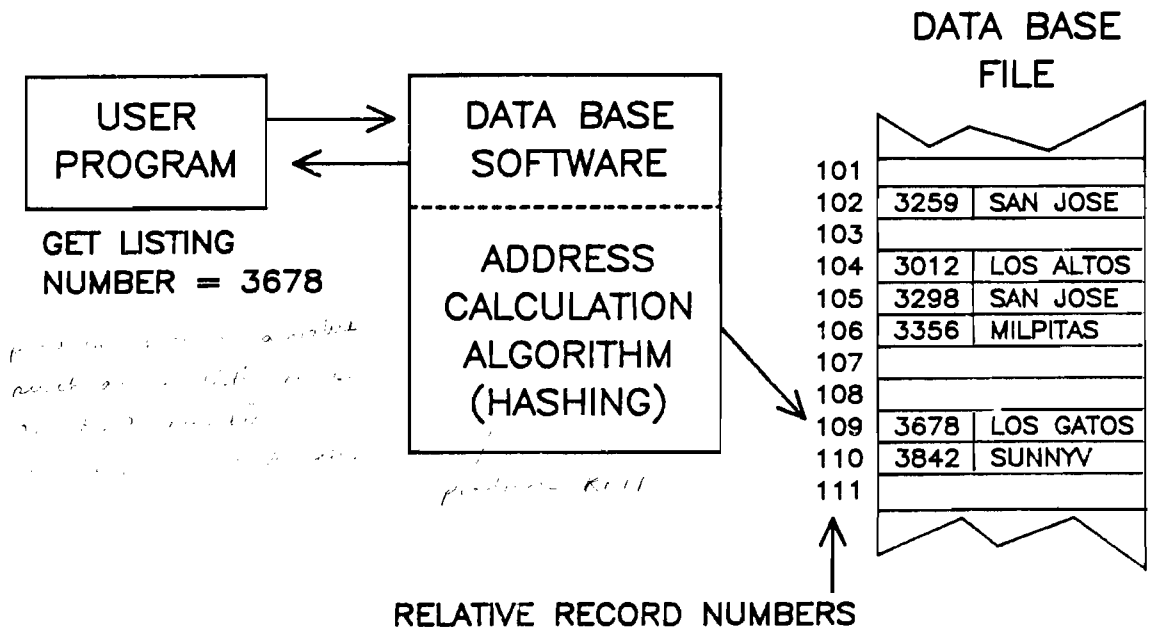


* ignores blank records
* begins at first or most recently accessed record - Forward

references: See slide IM1 3.04 - Serial & Random Access.

DATA BASE ACCESS

* CALCULATED



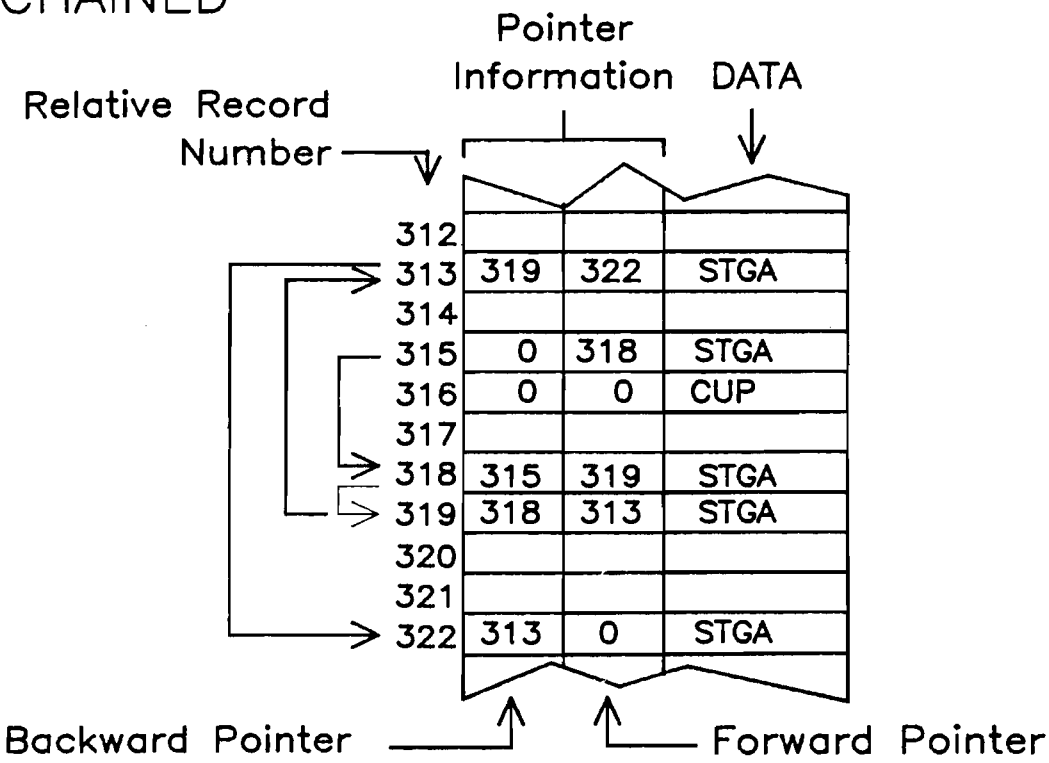
notes:

- calculated access and hashed access are synonomous

references:

DATA BASE ACCESS

* CHAINED



notes:

- observe that chained and calculated access are actually variations of the random access method.

references:

WHAT

IS

DATA BASE
? ? ?

notes:

- what does 'DBMS' (Data Base Management System) mean?

references:

DATA BASE

A collection of interrelated data that can be accessed by one or more application systems.

notes:

references:

DBMS
DATA BASE MANAGEMENT SYSTEM

A collection of software routines for:

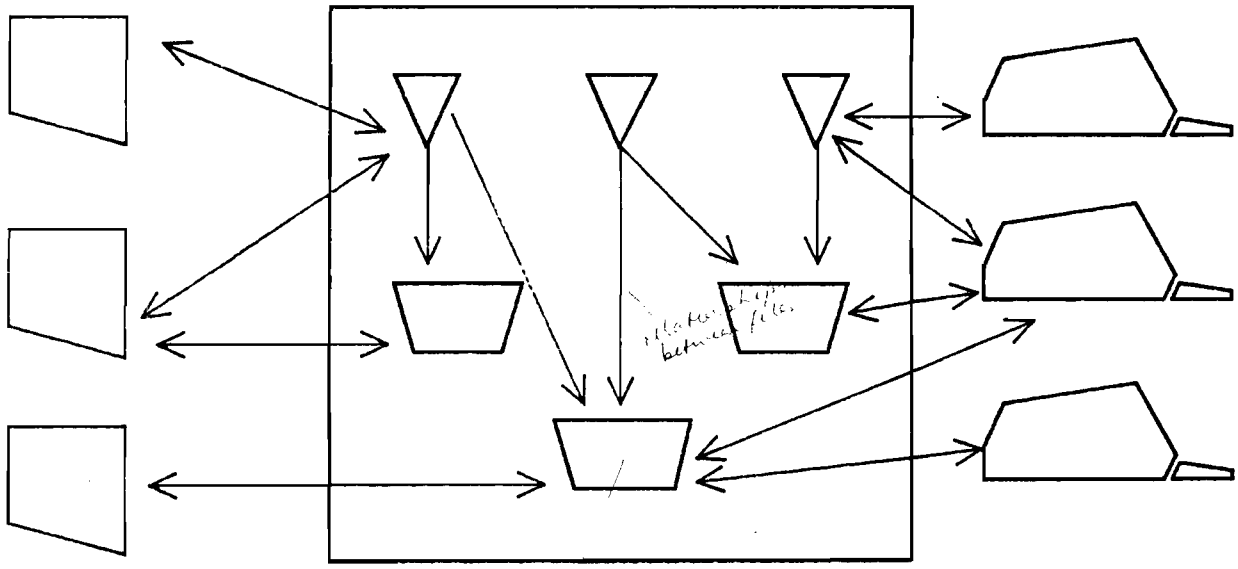
- Data Definition
- Privacy and Security
- Data Modification/Retrieval
- Backup and Recovery
- Maintenance
- Inquiry

notes:

- DBMS is the interface between application program and the data base

references:

DATA BASE



BATCH PROGRAMS

A collection of interrelated data stored on DASD

INTERACTIVE USERS

notes:

- DASD stands for Direct Access Storage Device (e.g. magnetic disc)

references:

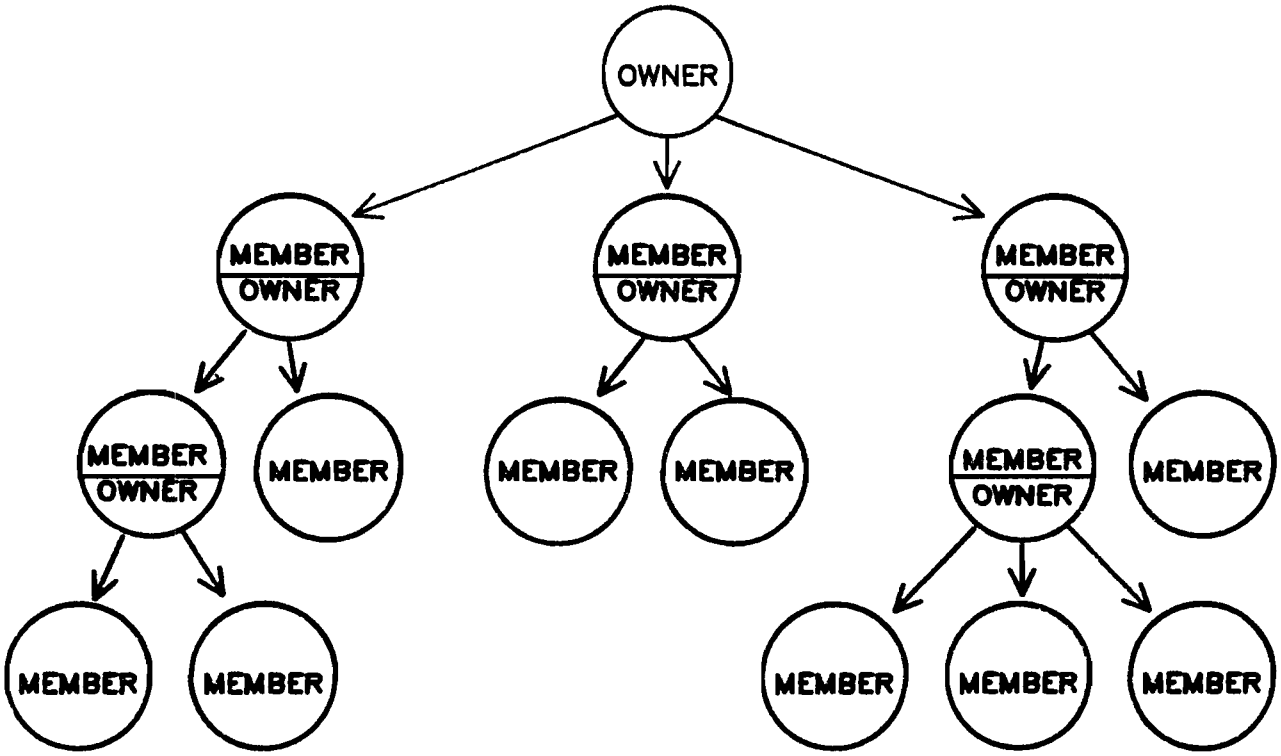
DATA BASE MODELS

* Hierarchical
* Network
* Relational

notes:

references:

HIERARCHICAL MODEL



Pre–defined relationships

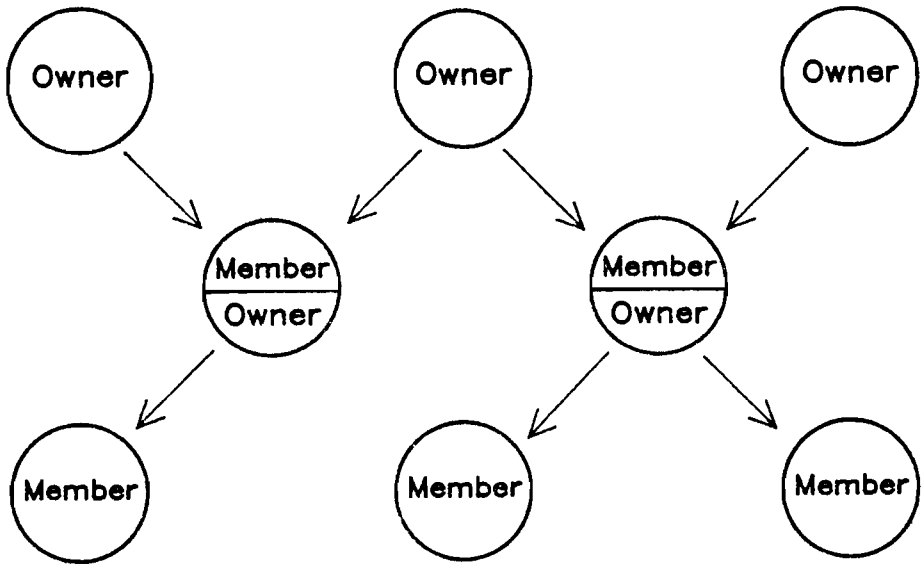
notes:

- a given member can have only one owner

- TREE structure — parent or owner may have many 'children or members'
  - member/children can only have ONE owner/parent
  - one to many relationship - like one realtor to many properties, one dept to many employees

references:

NETWORK MODEL

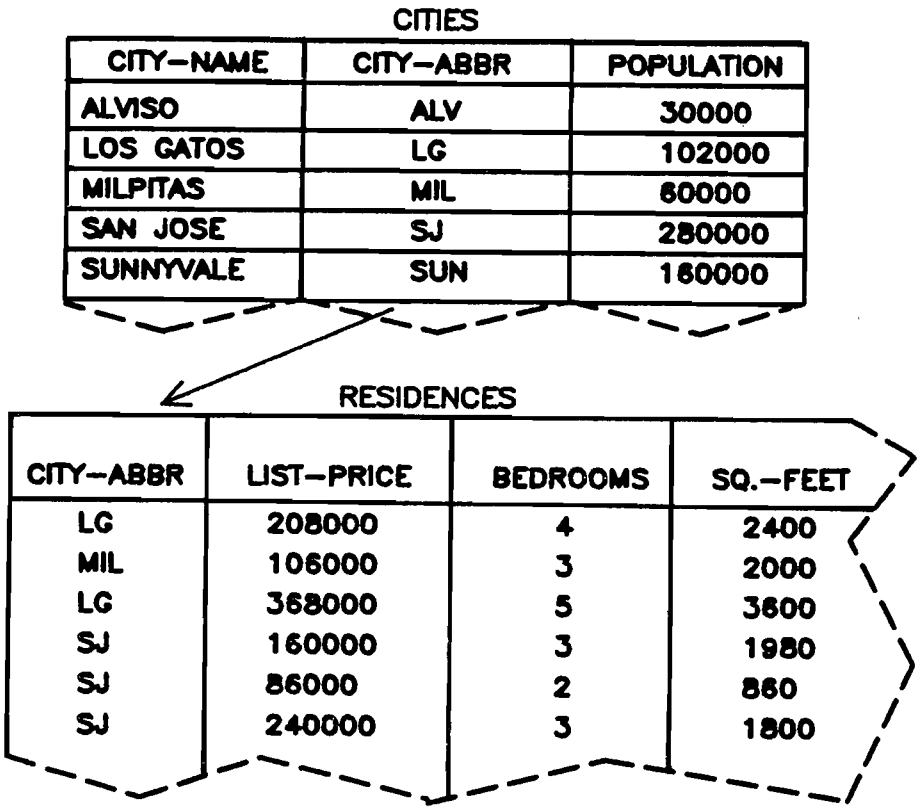


Pre–defined relationships

notes:

references:

RELATIONAL MODEL



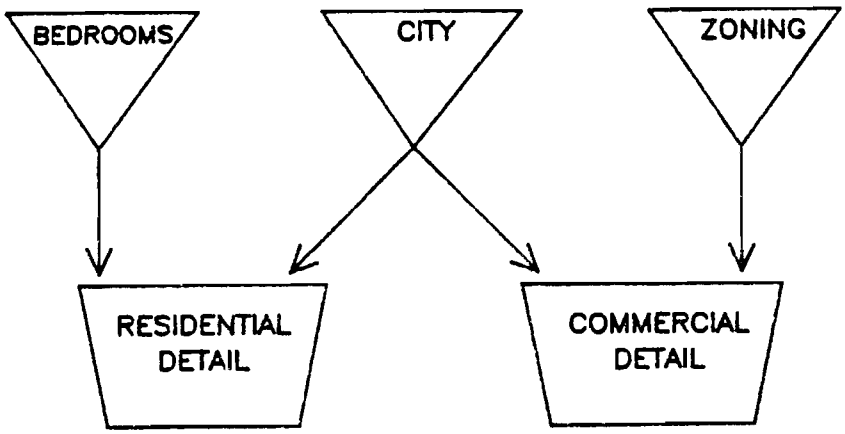
Relationships established at inquiry

notes:

- relations are sets of 2-dimentional tables
-

references:

IMAGE IS A NETWORK STRUCTURE



notes:

- a 2-level network structure

references:

DBMS SOFTWARE USER RATINGS

BY

DATAPRO RESEARCH CORP.

VENDOR	PRODUCT	No. Responses	Overall Satisfaction	Reliability	Efficiency	Ease of Installation	Ease of Use	Troubleshooting	Documentation	User Education	Vendor Maint.
HEWLETT-PACKARD	IMAGE/3000 N	27	3.4	3.7	3.1	3.6	3.5	3.0	2.9	2.8	3.3
MATHEMATICA	RAMIS II H	22	3.4	3.5	2.9	3.4	3.6	3.0	3.1	3.3	3.2
CULLINANE CORP	IDMS N	52	3.3	3.6	3.2	3.3	3.2	3.3	3.1	3.3	3.2
SOFTWARE AG	ADABAS	41	3.2	3.6	3.2	3.2	3.4	2.8	2.8	2.8	2.9
INFO BUILDERS	FOCUS R	21	3.1	3.3	2.9	3.3	3.6	2.8	2.8	3.0	3.1
IBM	IMS H	18	3.1	3.3	2.7	2.2	2.7	2.8	2.7	2.9	3.1
HEWLETT-PACKARD	IMAGE/1000 N	04	3.0	3.5	3.0	3.0	3.8	3.0	2.8	2.8	3.0
CINCOM	TOTAL N	54	3.0	3.4	2.9	3.0	3.0	2.6	2.7	2.8	2.8
INTEL	SYSTEM 2000 H	12	3.0	3.6	2.6	2.9	3.5	2.9	2.6	3.0	3.3
BURROUGHS	DMS II N, H	29	3.0	3.5	3.1	3.1	3.3	2.2	2.4	2.4	2.7
COMPUTER CO. OF AM.	MODEL 204	06	2.8	3.2	3.6	3.3	3.4	2.2	2.6	2.6	2.6
DIGITAL EQUIP	DBMS	04	2.8	3.0	2.5	2.8	2.8	2.0	2.8	2.8	3.0
TANDEM	ENFORM	03	2.7	3.7	2.0	3.3	3.0	2.5	2.3	1.7	3.0
PRIME	(NO ENTRY)										

Source: COMPUTERWORLD, December 21,1981

notes:

references:

WHY

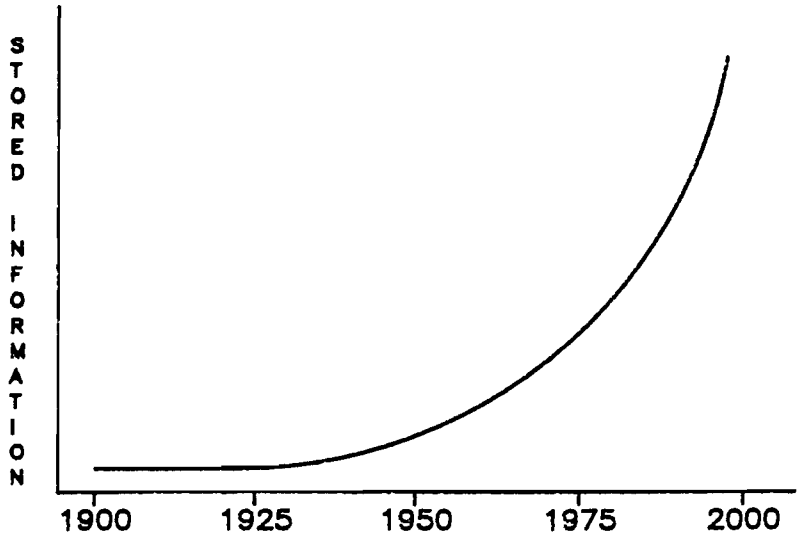
DATA BASE

? ? ?

notes:

references:

INFORMATION MANAGEMENT CHALLENGE

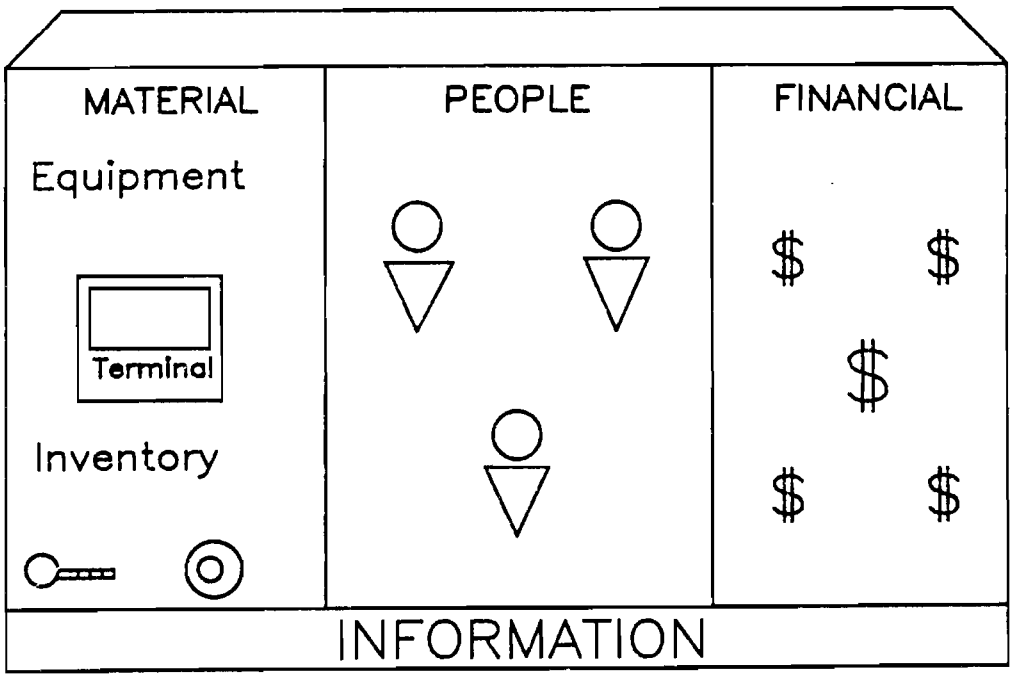


The proper management of too much data

notes:

references:

COMPANY ASSETS

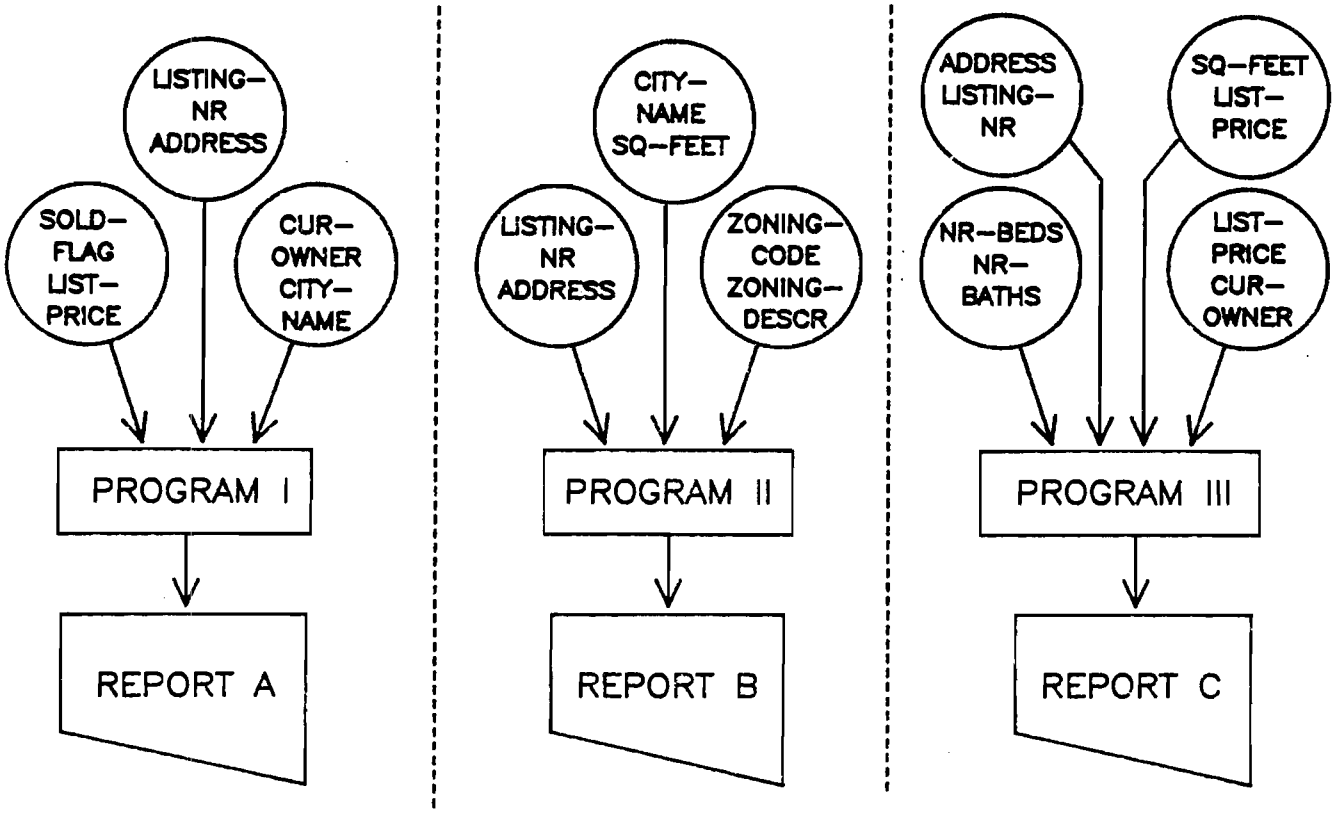


* Planning * Managing * Controlling

notes:

references:

TRADITIONAL APPROACH TO INFORMATION MANAGEMENT



notes:

references:

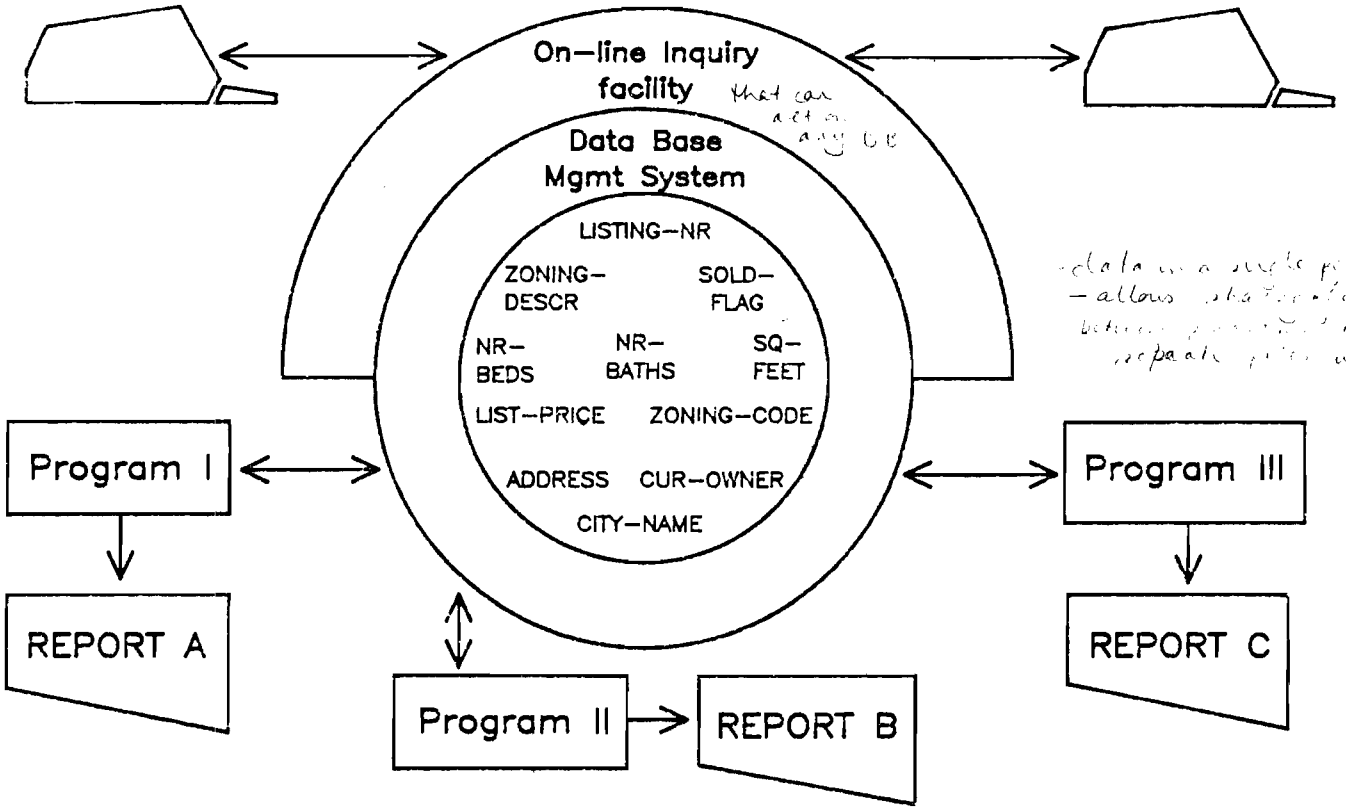
BEFORE DATA BASE

* Programs dependent on file form and content
* Data redundancy high
* Programs 'owned' data — change difficult and expensive
* High program maintenance costs
* Data inconsistency
* Lack of data control
* Sharing data difficult

notes:

references:

DATA BASE APPROACH TO INFORMATION MANAGEMENT



notes:

references:

DATA BASE BECAUSE:

* Centralized control over data
  - Reduces redundancy
  - Avoids inconsistencies
  - Sharable data
  - Enforceable standards
  - Greater security
* Reduce application development time
* Reduce effects of change
* Lower program maintenance costs
* Alternate views of data available

notes:

references:

IMAGE/3000

DBMS
OVERVIEW

ch 2 - IMAGE

notes:

references:

IMAGE TERMS

DATA ITEM – Smallest accessible element of information (FIELD)

DATA ENTRY – An ordered collection of related data items (RECORD)

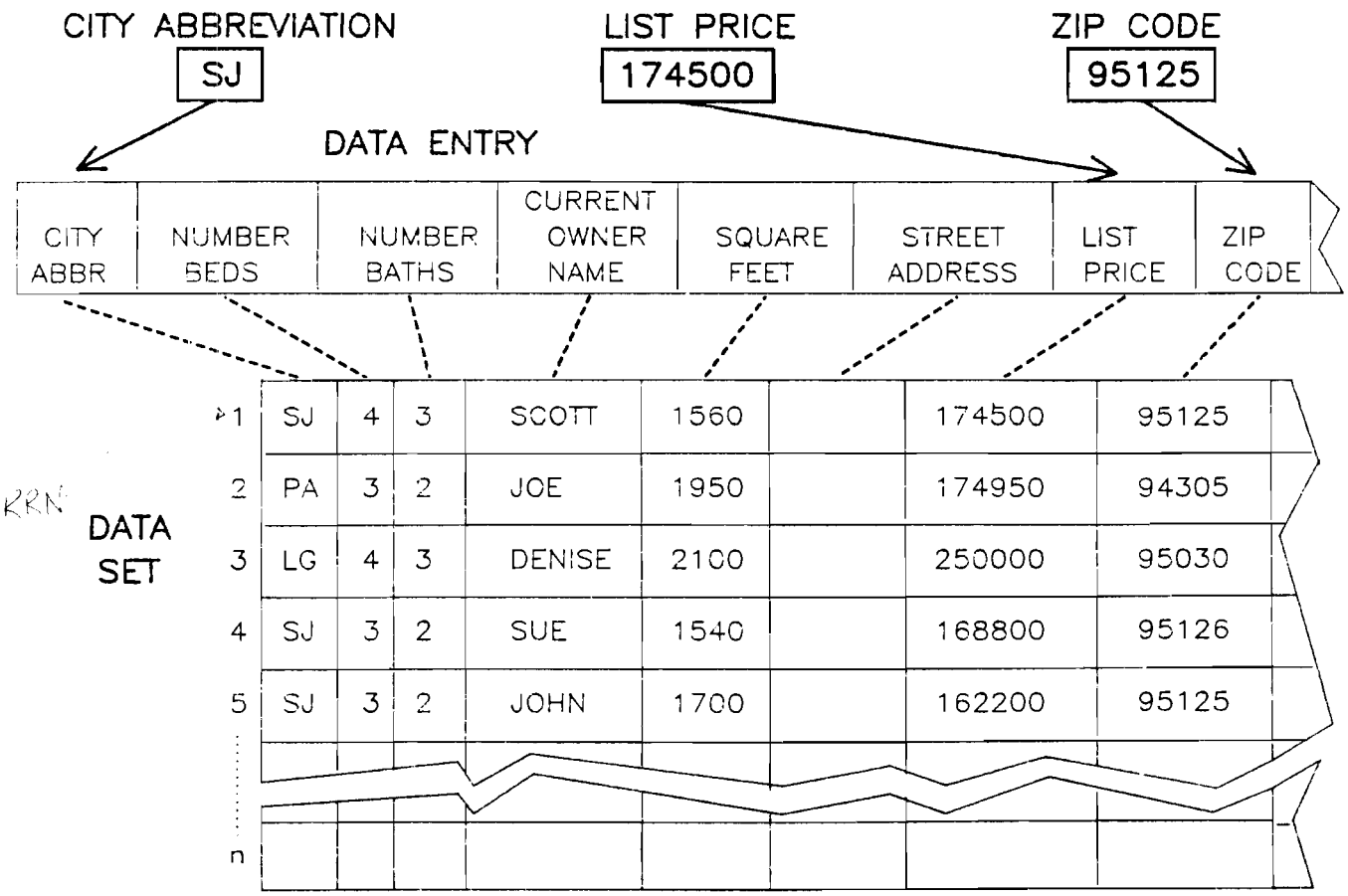
DATA SET – A collection of data entries sharing a common definition (FILE)

DATA BASE – A named collection of data sets

notes:

references:

DATA ITEMS



KRN

IM1 4.02

HEWLETT PACKARD

notes:

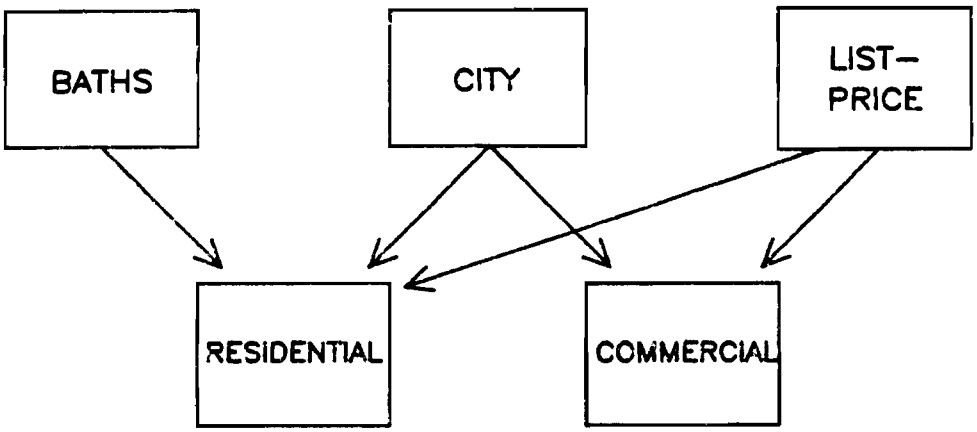
ITEMS are [illegible]
- [illegible] element of [illegible] data
- an [illegible] of [illegible] data

ENTRY - [illegible]

SET - [illegible] relation
to the [illegible] file - KRN

references:

DATA BASE



INTERRELATED DATA SETS

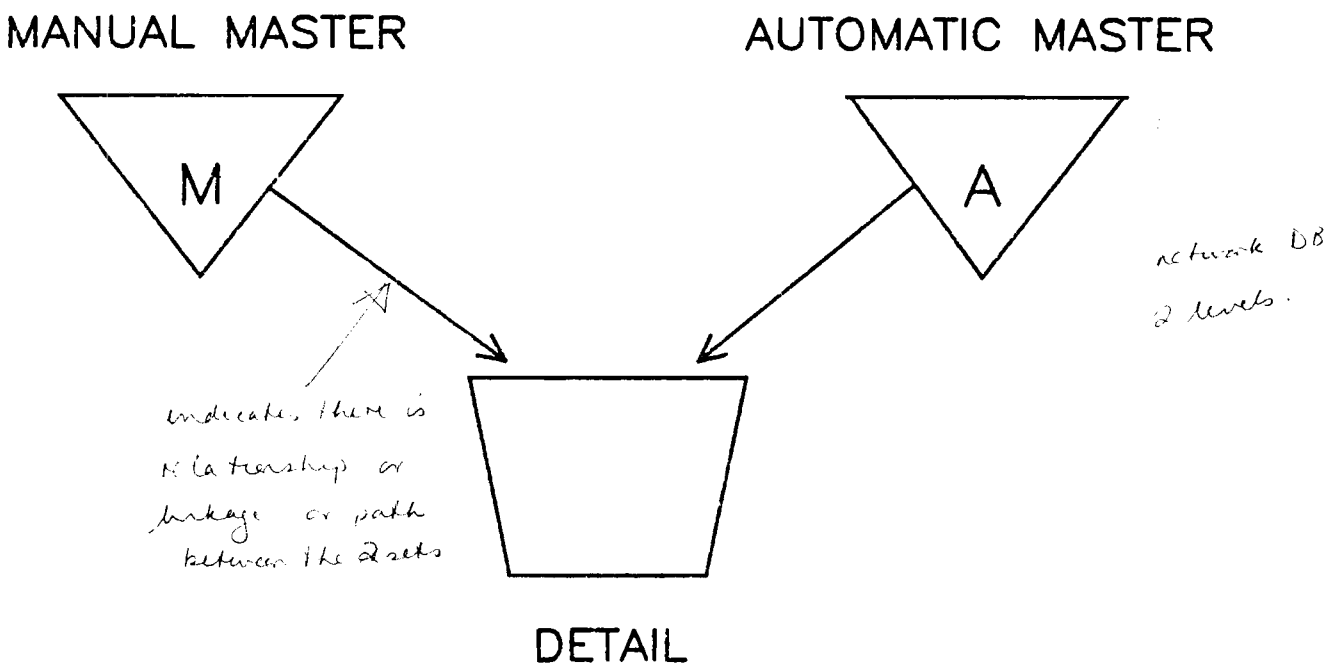
notes:

- each IMAGE data set is an MPE file

references:

IMAGE manual ch. ... 2-1 - 2-16

IMAGE DATA SETS



notes:

3 different types of data sets

▽ => master data set { Manual / Automatic

▽ => detail data set

references:

MASTER DATA SETS

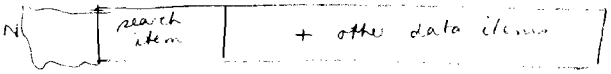
* Serve as index to related detail data sets
* Contain one search item (key) which must be a unique value in the data set
* May be related to up to 16 detail data sets
* Location of an entry in a master set is determined by passing the entry's search item (key value) through an address calculation algorithm (hashing) whose result is a relative record location within the master set

notes:

references:

MASTER DATA SETS
MANUAL

* All entries must be explicitly (manually) added or deleted by the user program
* May contain data items in addition to the key item



* A master entry must exist before a related detail entry with a matching key can be added
* Key values of existing master entries serve as a table of legitimate search item values for all related detail data sets
* Need not be related to detail sets (standalone)

notes:

references:

MASTER DATA SETS
AUTOMATIC

* All entries are implicitly (automatically) added or deleted as related detail entries are added or deleted
* Must be related to one or more detail data sets (cannot be standalone)
* Must contain only one data item: the search item (key)

notes:

references:

DETAIL DATA SETS

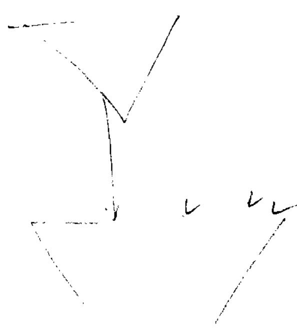
— allow 'like' entries to be retrieved quickly

* Entries may contain duplicate search item values
* Entries with duplicate search item values are 'chained' together with pointers
* Entries in a chain may be retrieved in sorted order
* May be related to up to 16 master data sets
* May be stand-alone — added flexibility

RRN | data entry

HEWLETT PACKARD

notes:



16 →

references:

IMAGE TERMINOLOGY

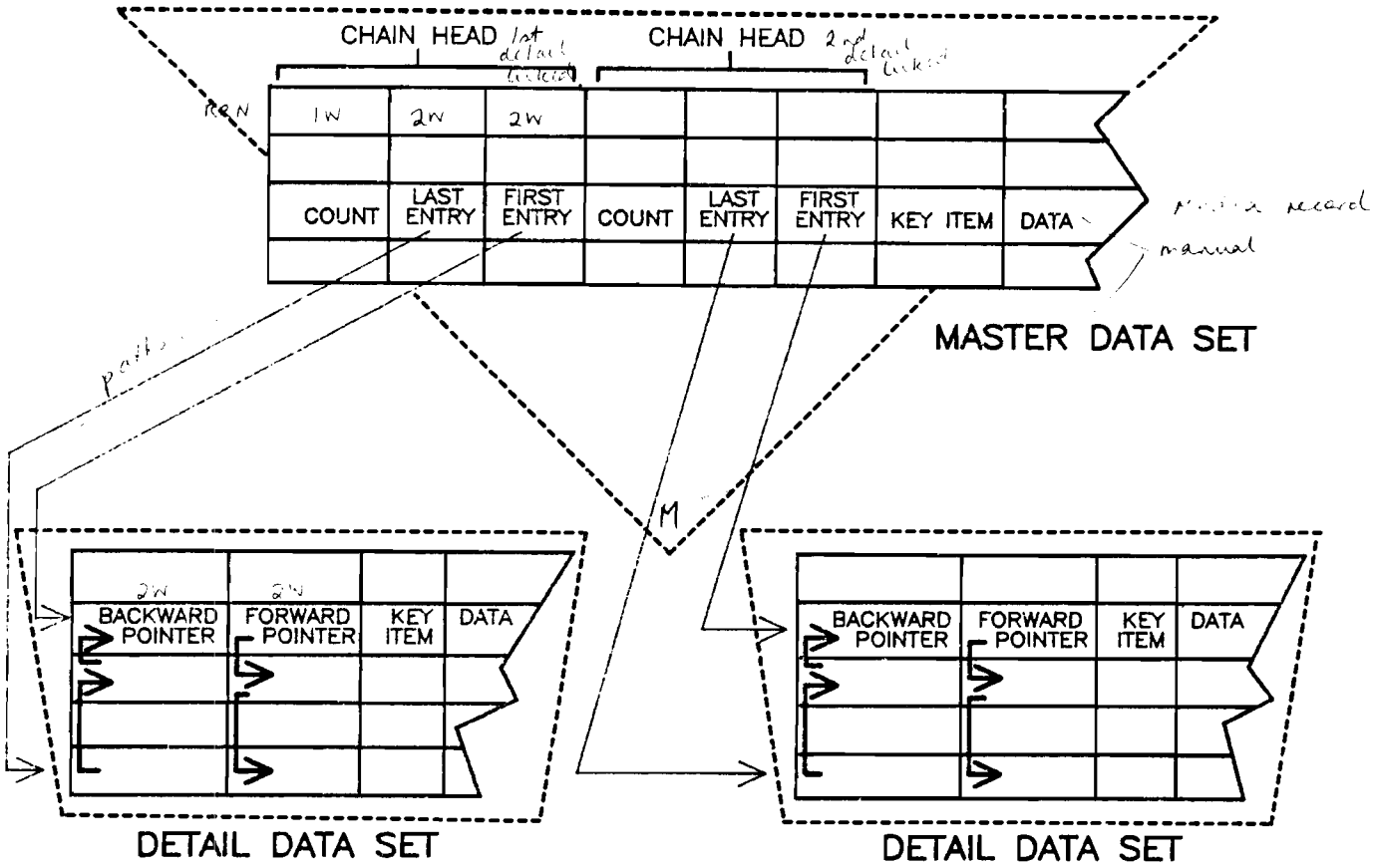
* RELATIVE RECORD NUMBER
  The location (address) of an entry (record) relative to its location from the beginning of a data set (file).
* POINTER
  The 'record address' of one entry stored in another entry. Access to one entry will allow direct access to the other.
* CHAIN
  Linkage of data entries via pointers.
* COUNT
  The number of entries in a chain.

notes:

- relative record number and pointer are a review from Data Management section
- in detail data sets, entries with duplicate search item values are 'chained' together

references:

ASSOCIATION OF IMAGE DATA SETS



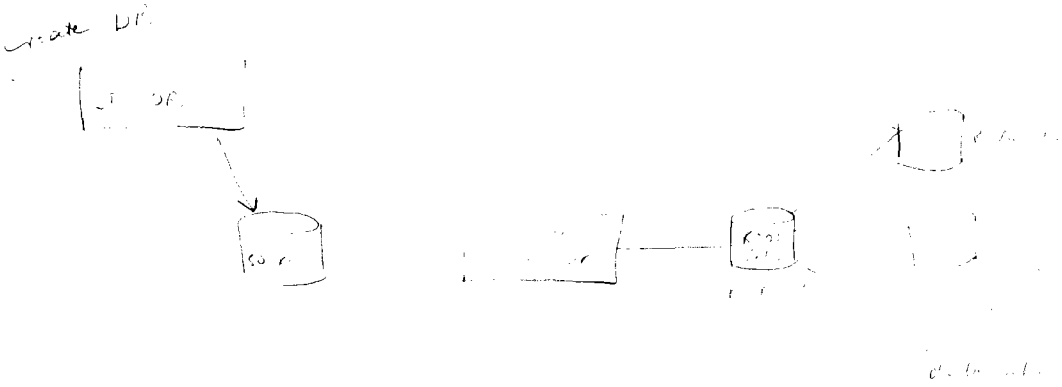
notes:

references:

COMPONENTS OF IMAGE

* Data Definition Language (DDL)
  - Used to define all aspects of the data base
  - Defines data items, security, relationships and capacities
* Data Manipulation Language
  - Provides the interface between application programs and the data base
  - A set of library routines (procedures), called from application programs

notes:



references:

COMPONENTS OF IMAGE

* Image Utilities
  - Utility programs for creating and maintaining data bases
  - Programs for back—up and recovery

* QUERY/3000
  - A program to access any data base
  - Allows inquiries and modification of data bases
  - Allows display of data base structure

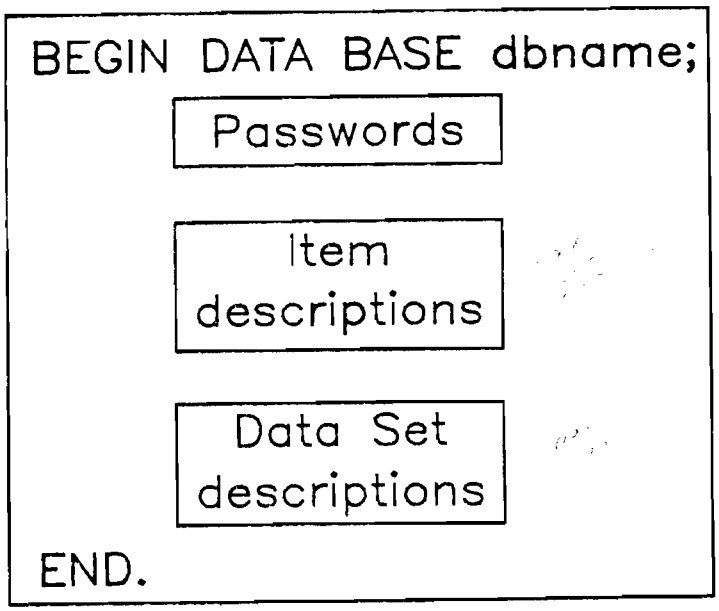
notes:

references:

DATA DEFINITION LANGUAGE

* Simple format
* Easy to understand

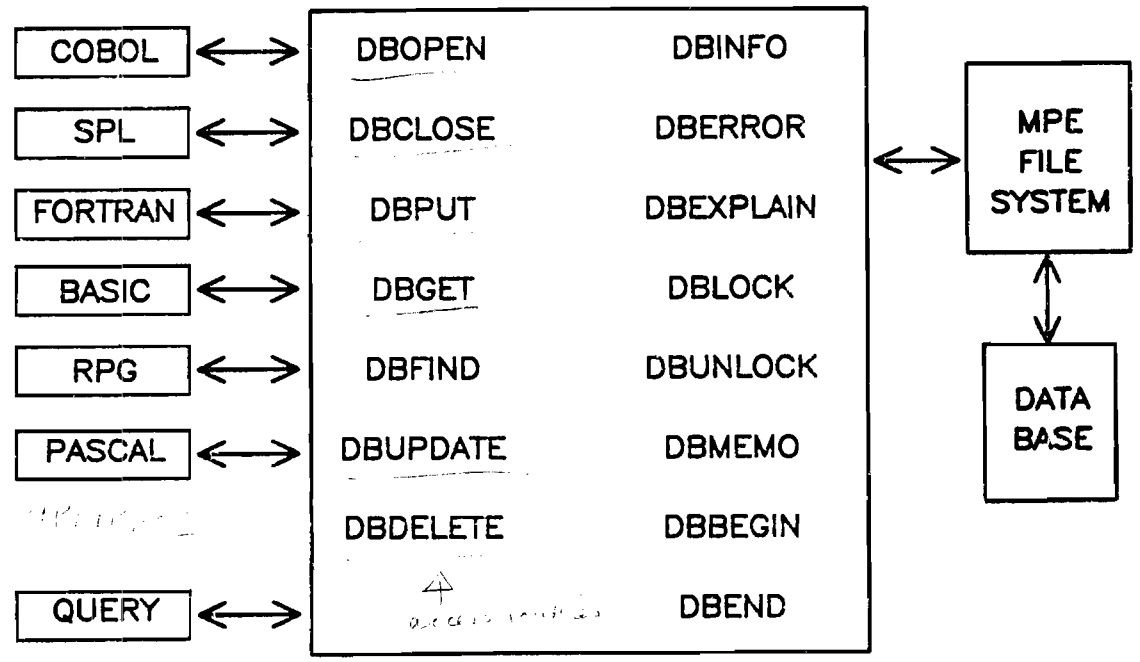
THE 'SCHEMA'



notes:

references:

DATA MANIPULATION LANGUAGE



HEWLETT PACKARD

notes:

references:

IMAGE UTILITIES

DBSTORE – Write complete data base to tape

DBRESTOR – Write data base from tape to disc

DBUNLOAD – Write data only to tape

DBLOAD – Write data into empty data base from tape

DBUTIL – Create, erase, and purge data base, set flags, show, etc.

DBRECOV – Write transactions to data base from logging file

notes:

references: MPE Pocket Guide, Section IV, IMAGE

LAB 1

DATA BASE OVERVIEW

notes:

references:

1. List the data base access methods.

2. Contrast serial and directed accesses.

3. What is the purpose of the address calculation algorithm?

4. What are the three data base models and which model is IMAGE?

5. Name at least three benefits of a data base.

6. Define the following IMAGE terms:

   a. Data item

   b. Data entry

   c. Data set

   d. Data base

7. Name the IMAGE data set types.

8. What are the differences between the two types of master data sets?

9. a. Describe the relationship between Automatic Master and Detail data sets.

   b. Can Manual masters have the same relationship? Explain.

10. a. What is the criteria for detail data entries being on the same chain?

    b. How are they linked together internally?

11. What is a relative record number?

12. What is a pointer?

13. Define a data chain. How does IMAGE know how many entries are in a data chain?

14. List the components of IMAGE and explain what they do.

15. List the IMAGE Utilities and explain what they do.

SOLUTION:
The solution of this lab can be found in LAB1.SOLUTION.

IMAGE/3000

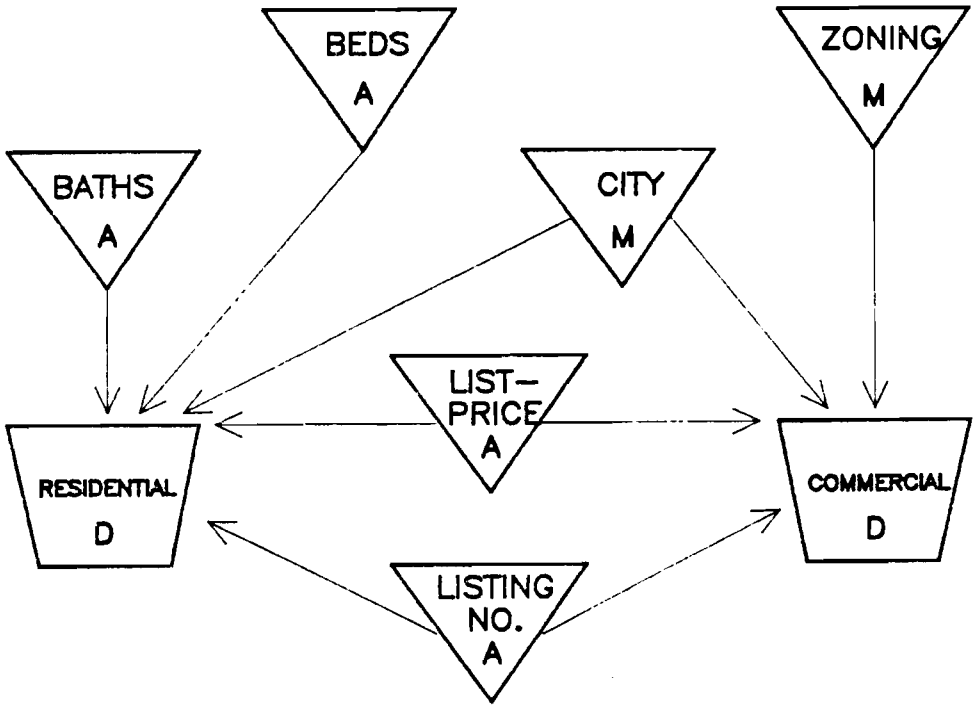
AND

THE CASE STUDY

notes:

references:

THE 'REALTY' DATA BASE



notes:

references:

LAB 2

INTRODUCTION TO

THE CASE STUDY

notes:

references:

LAB 2 -- INTRODUCTION TO THE CASE STUDY

1. INTRODUCTION
   You are a salesperson at Wonder Realty and you have decided to list your own house in San Jose.

2. OBJECTIVE
   Gain familiarity with the REALTY data base and the lab program that will be developed piecemeal throughout the rest of the course.

3. INSTRUCTIONS
   A. FILES: REALTY.LABS — Realty data base
      LABDEMO.LABS — Lab demonstration program

   B. DETAILED INSTRUCTIONS:

      1. Log on to the system using the provided user and account names.

      2. Run LABDEMO.LABS

      3. Respond to the question "Which lab are you running?" with the lab number: "2", in this case. Use "MANAGER" (Note: Upper case) as your password.

      4. Add your present dwelling to the REALTY data base.

         a. Indicate that you want to work with the residential listing information.

         b. On the residential listing information screen, enter the appropriate function to add an entry.

         c. Fill all the fields (making up your data as needed). Use "SJ" for CITY-ABBREV; NUMBER OF BATHS should be in the form X.XX. Note that the LISTING-NR (which is the only unique key) will be automatically provided.

      5. Find/list all houses in San Jose to compare list prices with yours.

      6. Find/list all houses with 3 bedrooms.

      7. Your phone just changed to 467-8309. Update your listing.

         a. Find your house entry.

         b. Update the entry by using the appropriate function and entering the new phone number.

      8. Feel free to browse through the application.

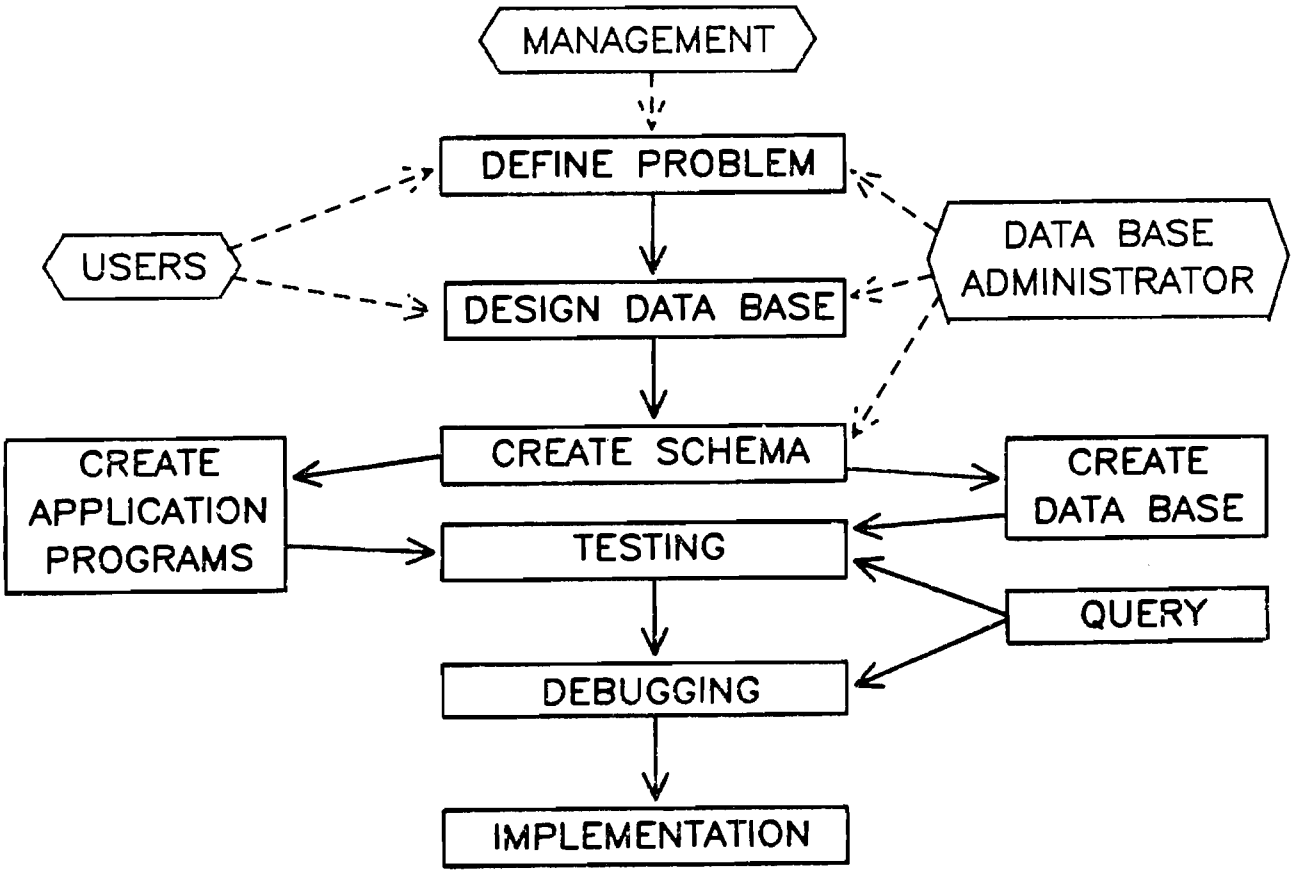
IMAGE IMPLEMENTATION:

AN OVERVIEW

notes:

references:

APPLICATION IMPLEMENTATION



notes:

references:

DEFINE PROBLEM

* Toughest part
* Requires management commitment
* Requires user involvement
* Longest segment of implementation cycle

notes:

references:

DESIGN DATA BASE

* Consider wants vs needs
* Consider performance
* Consider future change
* Draw data base flowchart
* More on this in section XIII

notes:

references:

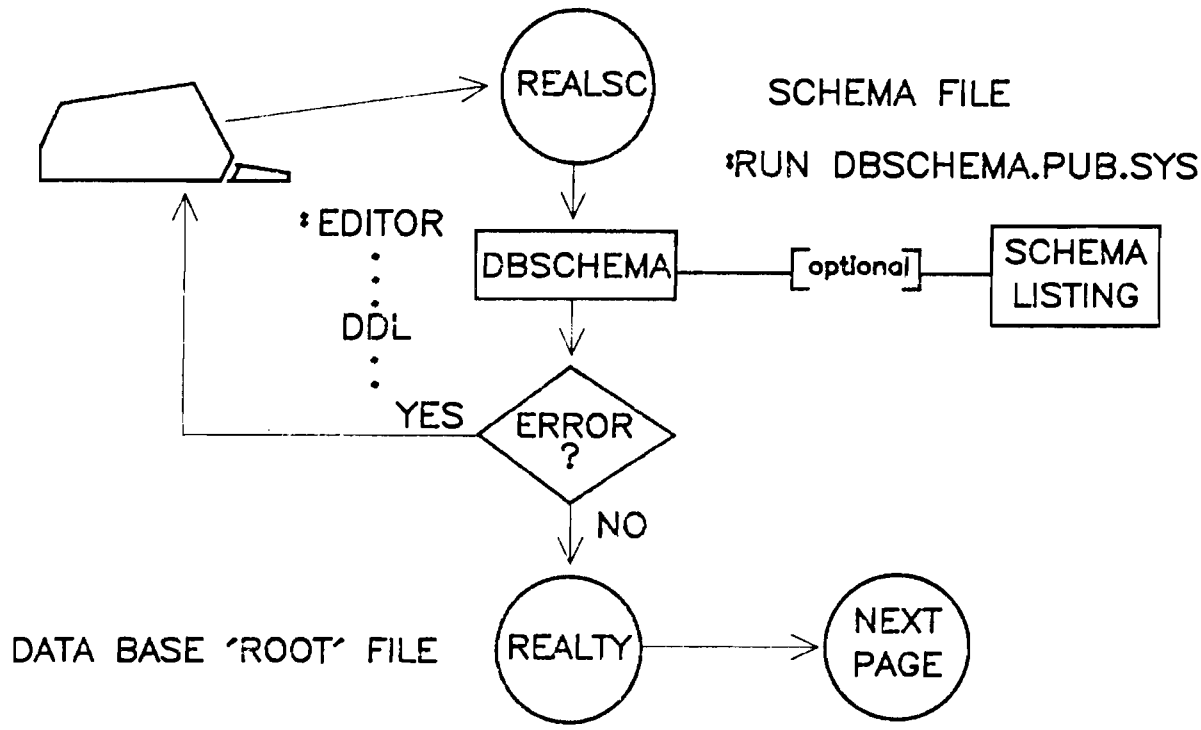
CREATE SCHEMA

* Define data item names
* Define data item types
* Define data item lengths
* Define data security
* Determine data set capacities
* With schema listing application program development can begin
* Details in section VIII

notes:

references:

CREATE DATA BASE

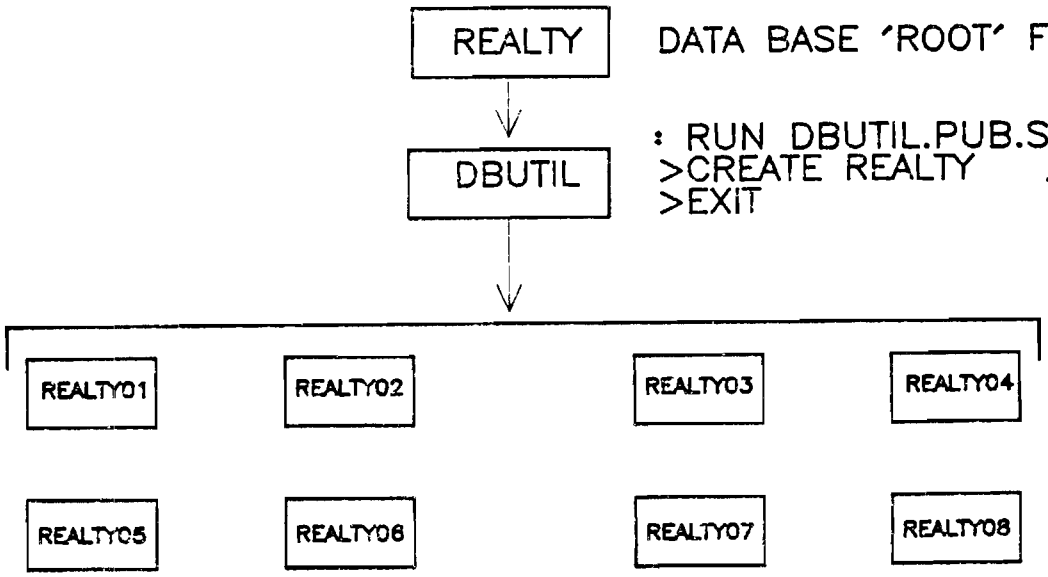


notes:

- running DBSCHEMA creates only the data base root file

references:

CREATE DATA BASE (cont'd)



— DATA BASE READY FOR USE —

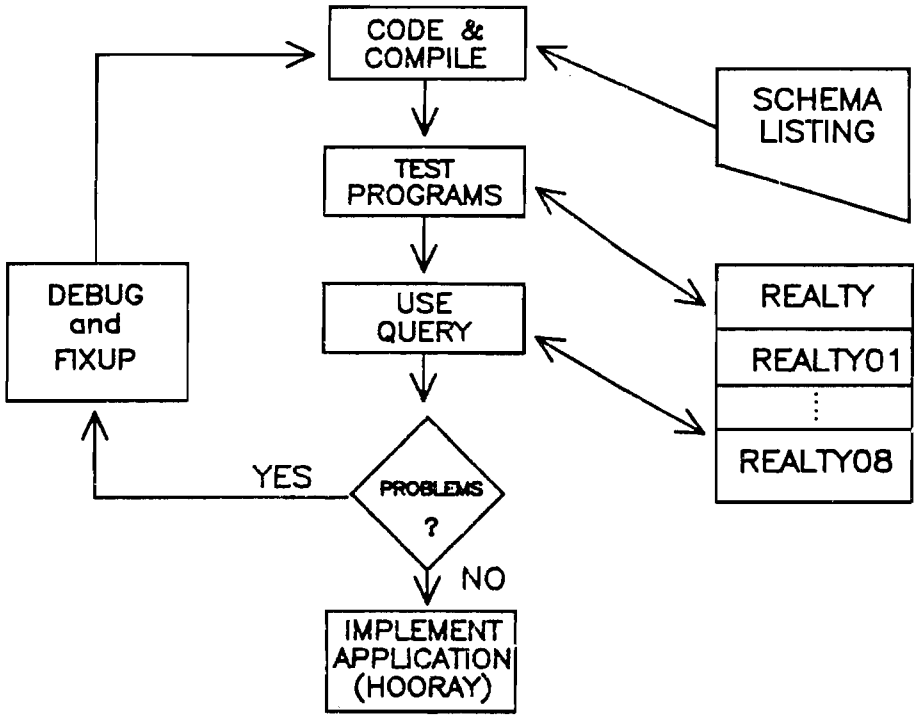
notes:

!LISTF

- data set names are REALTY01 - REALTY08. Data sets are numbered according to their appearance in the Schema.

references:

APPLICATION PROGRAM DEVELOPMENT



notes:

- program development can begin once the data base has been designed and defined.

references:

IMAGE IMPLEMENTATION

ADDITIONAL CONSIDERATIONS

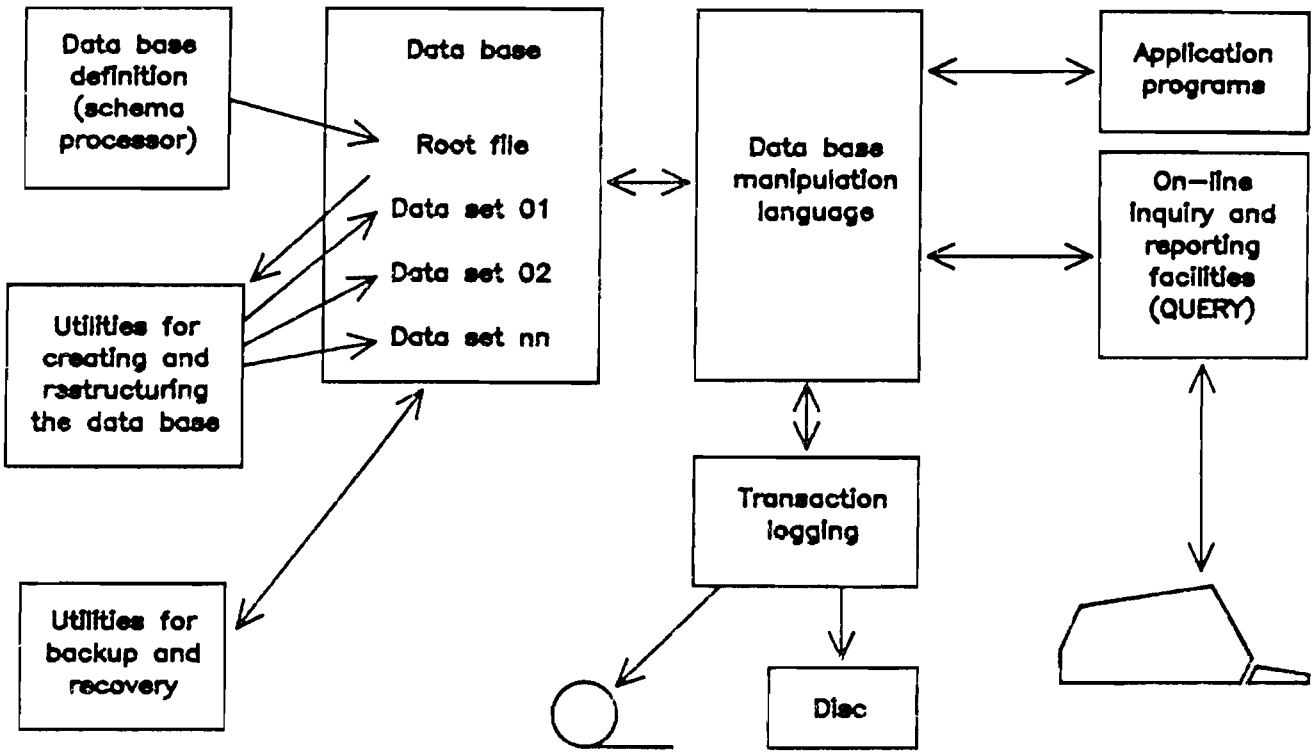
* Multi-user access control (Locking)
* Transaction logging
* Backup strategy
* Recovery scheme

notes:

- these are important application design and operational considerations. Details in sections 9 and 10.

references:

IMAGE OVERVIEW



notes:

references:

QUERY/3000

notes:

references:

QUERY/3000

* A powerful program development and debugging tool

* This is a short introduction

notes:

references:

QUERY APPLICATIONS

* Casual inquiry of the data base

* Data base modification (low volume)
  - Additions
  - Deletions
  - Updates

* Report generation

* Application program development
  - Testing data base designs
  - Debugging programs

notes:

references:

COMMANDS
FOR PROGRAM DEVELOPMENT

> HELP
> DEFINE
> FORM
> FIND
> REPORT ALL
> LIST
> ADD
> OUTPUT

notes:

references:

INVOKING QUERY

```
:RUN QUERY.PUB.SYS
.
.
.
QUERY/3000 READY

>
```


notes:

references:

SPECIAL CHARACTERS

>	Prompt for QUERY command
>>	Prompt for additional information
&	Line continuation (Must be followed directly by C/R)
^C	(CNTL Y) – Terminates the QUERY command, prints <CONTROL Y>, and prompts for a new command

notes:

references:

> HELP Defines and describes QUERY commands

SYNTAX:

H[ELP] [command name]

notes:

references:

> DEFINE

To identify a particular data base and its attributes to QUERY

SYNTAX:

DEF[INE]
- data base
- password
- mode
- data-sets
- proc-file
- output

notes:

references:

EXAMPLE:

```
> DEFINE
    DATA-BASE = >> REALTY
    PASSWORD = >>
    MODE = >> 1
    DATA-SETS = >>  (C/R)
    PROC-FILE = >>  (C/R)
    OUTPUT = TERM
    OUTPUT = >>  (C/R)
>
```


notes:

- all user input is underlined
- (C/R) = Carriage Return
- password will not be echoed

references:

> FORM Displays the data base structure

SYNTAX:

FO[RM] [SETS]
[ITEMS]
[PATHS]
•
•
•

notes:

- will only give you what you have access to
- very similar to the schema
- try it in the lab

references:

FORM EXAMPLE:

>FO SETS

DATA BASE: REALTY

SETS:	TYPE	ITEM COUNT	CAPACITY	ENTRY COUNT	ENTRY LENGTH	BLOCKING FACTOR
LISTNR-MASTER	A	1	61	50	1	7
LIST-PRICE-MSTR	A	1	61	42	2	7
BATH-MASTER	A	1	61	2	2	9
BEDS-MASTER	A	1	61	4	1	11
ZONING-MASTER	M	2	13	6	12	5
CITY-MASTER	M	2	41	26	12	4
RESIDENTIAL	D	12	63	40	47	9
COMMERCIAL	D	10	65	10	32	13

notes:

references:

FORM EXAMPLE:

```
>FORM RESIDENTIAL

DATA BASE: REALTY

SET NAME:
   RESIDENTIAL,DETAIL

      ITEMS:
         LISTING-NR,              I1      <<SEARCH ITEM>>
         CITY-ABBR,               X4      <<SEARCH ITEM>>
         LIST-PRICE,              I2      <<SEARCH ITEM>>
         NUMBER-BEDS,             X2      <<SEARCH ITEM>>
         NUMBER-BATHS,            X4      <<SEARCH ITEM>>
         CURRENT-OWNER,           X20
         OWNERS-PHONENR,          X10
         STREET-ADDRESS,          X30
         ZIP-CODE,                X6
         FORMAL-DINING-RM,        X2
         SOLD-FLAG,               X2
         SQUARE-FEET,             X8      <<SORT ITEM>>
```


notes:

references:

> FIND

To selectively retrieve entries in data base

SYNTAX:

F[IND] [data set.] data item

⟨RELATIONAL OPERATOR⟩ value

[,value...] [AND / OR] ...[END]

notes:

references: See MPE Pocket Guide Section IV (QUERY) for a list of available relational operators.

EXAMPLE:

> FIND

PROBLEM:

A user wants to select all the residences less than $150,000 with between 1600 and 2000 sq. feet, 3 bedrooms, and 2 or more bathrooms.

QUERY SOLUTION:

```
> FIND LIST-PRICE ILT 150000 AND &
> SQUARE-FEET IB 1600,2000 AND &
> NUMBER-BEDS EQ 3 AND &
> NUMBER-BATHS >= 2
6 ENTRIES QUALIFIED
```


notes:

references:

> FIND ALL Locates all entries in the specified data set

SYNTAX:

F[IND] ALL [data set.] data item

EXAMPLE:

FIND ALL RESIDENCES.LIST-PRICE

notes:

references:

> REPORT ALL

Produces an unformatted report of those data items retrieved by the last FIND command

SYNTAX:

R[EPORT] ALL [,character]

WHERE character IS ANY CHARACTER.
IF INCLUDED, data item names ARE OMITTED.

notes:

references:

REPORT ALL EXAMPLE:

```
>FIND CITY-ABBR IS CAP
1   ENTRIES QUALIFIED
>R ALL


LISTING-NR              =12366
CITY-ABBR               =CAP
LIST-PRICE              =198000
NUMBER-BEDS             =2
NUMBER-BATHS            =2
CURRENT-OWNER           =JOSEPH DOAKS
OWNERS-PHONENR          =4084768771
STREET-ADDRESS          =324 EAST CLIFF DRIVE
ZIP-CODE                =95062
FORMAL-DINING-RM        =NO
SOLD-FLAG               =
SQUARE-FEET             =1460

>
```


notes:

- FIND must precede REPORT

references:

> LIST

Prints complete or partial entries from a single data set with automatic formatting and headings

SYNTAX:

L[IST] ⟨data set name / data item list⟩

[FOR data item relop "value"

[AND / OR ...]

notes:

- combines FIND and REPORT commands
- truncates at column 80 on terminals

references:

LIST EXAMPLE:

```
>LIST RESIDENTIAL

LISTIN  CITY  LIST-PRICE  NU  NUMB  CURRENT-OWNER       OWNERS-PHO
     1  ALMD      175000  03  2.00  WILLIAM JONES       4153851600
     2  ALMD      258500  03  2.00  CHAGE WARREN        4153854425
     3  BV        181900  04  2.00  BOB CASKEY          4154297033
     4  BV        240000  03  2.00  DENISE WELCH        4154299877
     5  CAMB      167000  04  2.00  BOB OAKS            4154220963
     6  CAMB      249950  05  3.00  DAVE DITCH          4154223435
     7  CMBL      179000  05  2.00  BECKY GOLDBERG      4153782667
     8  CMBL      314950  06  3.00  JACK SLOCUM         4153785690
     9  CUP       177500  04  2.00  HAL SMITH           4152521964
    10  CUP       289500  04  2.00  BILL POWERS         4152523357
 12366  CAP       198000  2   2     JOSEPH DOAKES       4084768771
    12  EV        175000  04  3.00  JILL SMITH SECOND   4157762534
    13  EVG       179500  04  3.00  BOB DESI
< CONTROL Y >

>
```

IM1 7.18 HEWLETT PACKARD

notes:

references:

> ADD Add data to data base

ADD EXAMPLE:

```
> ADD RESIDENTIAL
LISTING-NR              =>> 12355
CITY-ABBR               =>> SJ
LIST-PRICE              =>> 105000
NUMBER-BEDS             =>> 3
NUMBER-BATHS            =>> 2.5
CURRENT-OWNER           =>> STEVE JOBS
OWNERS-PHONENR          =>> 4089965555
STREET-ADDRESS          =>> 122 LIBERTY ST.
ZIP-CODE                =>> 95040
FORMAL-DINING-RM        =>> NO
SOLD-FLAG               =>>
SQUARE-FEET             =>> 1780

LISTING-NR              =>> //
>
```


notes:

references:

> OUTPUT

Designates the destination of all output

SYNTAX:

OUT[PUT] = [TERM / LP]

notes:

- if OUTPUT=LP is used, reports will not print until QUERY is EXITed, or OUT=TERM is specified
- no :FILE equation necessary

references:

> EXIT Terminates QUERY; returns control to MPE

SYNTAX:

E[XIT]

notes:

references:

LAB 3

QUERY COMMANDS

notes:

references:

1. Log on to the system using the user and account names supplied by the instructor.

2. Run QUERY and use the HELP command to look at QUERY commands. Use HELP FORM to see the options of the FORM command.

3. Define your environment as follows:
   DATA BASE = REALTY.LABS
   PASSWORD = RECEPT
   MODE = 1

4. Enter a FORM command and view the results. Try other options of the FORM command.

5. List all the QUERY commands to the line printer. (Hint: Switch the OUTPUT)

6. List all the data sets and data items to the line printer.

7. List all entries in the RESIDENTIAL data set to the line printer in both possible ways.

8. Find all residences with prices between $180K and $190K, and list them to your terminal. (You can pick up your printout as soon as you switch the OUTPUT.)

9. Find all residences with three or more baths, and list them to your terminal.

10. Find all residences with prices between $180K and $190K, and that have three ("3.00") baths. List them to your terminal.

11. Exit QUERY, and log off the system.

SOLUTION:
The solution to this lab can be found in LAB3.SOLUTION.
Please refer to this AFTER you have attempted the lab on your own.

DATA BASE

CREATION

notes:

references:

TOPICS

* Data Definition Language
* Schema Processor (DBSCHEMA)
* Data Base Utility (DBUTIL)

notes:

references:

DATA

DEFINITION

LANGUAGE

Ch. 3 - [illegible] 3-1 → 3-25

notes:

references:

SCHEMA

* External description of database design
* Text editor MPE file
* Input to schema processor

notes:

references:

SCHEMA STRUCTURE

BEGIN DATA BASE data-base-name;

PASSWORDS: password-part;

ITEMS: item-part;

SETS: set-part;

END.

HEWLETT PACKARD

notes:

- columns 1 - 72 are processed only
- punctuation is required
- all alphabetic input to the Schema Processor is upshifted

references: IMAGE Reference Manual, Section III, Schema Structure

DATA BASE NAME

```
BEGIN DATA BASE REALTY;
```

* data base name may be up to 6 alphanumeric characters
  - First character must be alphabetic

notes:

```
BEGIN DATA BASE  data base name;
```

references:

PASSWORD PART

```
PASSWORDS:

    10          RECEPT;          << RECEPTIONIST          >>
    20          SALESREP;        << SALES PERSON          >>
    30          MANAGER;         << BOSS PERSON           >>
```

* Restricts access to data items and data sets
* user class number may be from 1 to 63
* password may be up to 8 ASCII characters
  - First character must be alphabetic

notes:

```
PASSWORDS:
  user class number [password];
```

- user class number must be unique within schema
- <<comments>> may appear anywhere

references:

IMAGE Reference Manual, Section II, User Classes and Passwords
Section III, Password Part

ITEM PART

```
ITEMS:
  CITY-ABBR,        X4    (10,20/30);   << CITY ABBREVIATION          >>
  CITY-NAME,        X20   (10,20/30);   << CITY NAME                  >>
  LIST-PRICE,       I1    (10,20/30);   << LIST PRICE TO NEAREST $1K  >>
  CURRENT-OWNER,    X20   (20/30);      << CURRENT OWNER              >>
  SOLD-FLAG,        X2    (10/20,30);   << "Y" OR BLANK              >>
  SQUARE-FEET,      X8    (10,20/30);   << SQUARE FEET                >>
```

* Defines each item in database
* item name may be up to 16 alphanumeric characters
  - First character must be alphabetic

notes:

```
ITEMS:
  item name, [sub-item count] type designator [sub-item length]
       [(read class list/write class list)];
```

references: IMAGE Reference Manual, Section II, Data Items
Section III, Item Part

ITEM PART

```
ITEMS:
    CITY-ABBR,          X4    (10,20/30):   << CITY ABBREVIATION          >>
    CITY-NAME,          X20   (10,20/30):   << CITY NAME                  >>
    LIST-PRICE,         I1    (10,20/30):   << LIST PRICE TO NEAREST $1K  >>
    CURRENT-OWNER,      X20   (20/30):      << CURRENT OWNER              >>
    SOLD-FLAG,          X2    (10/20,30):   << "Y" OR BLANK              >>
    SQUARE-FEET,        X8    (10,20/30):   << SQUARE FEET                >>

    MONTHS,         12  X4    (10,20/30):   << MONTHS                     >>
```

* sub-item count denotes occurrences of a sub-item within an item

  - May be from 1 to 255
  - Array type item (COBOL "OCCURS")
  - If omitted 1 is implied

notes:

```
ITEMS:
  item name, [sub-item count] type designator [sub-item length]
       [(read class list/write class list)];
```

references:

ITEM PART

```
ITEMS:
    CITY-ABBR,        X 4    (10,20/30);   << CITY ABBREVIATION          >>
    CITY-NAME,        X 20   (10,20/30);   << CITY NAME                  >>
    LIST-PRICE,       I 1    (10,20/30);   << LIST PRICE TO NEAREST $1K  >>
    CURRENT-OWNER,    X 20   (20/30);      << CURRENT OWNER               >>
    SOLD-FLAG,        X 2    (10/20,30);   << "Y" OR BLANK              >>
    SQUARE-FEET,      X 8    (10,20/30);   << SQUARE FEET                >>
```

* type designator denotes how the item value is stored on disc

notes:

```
ITEMS:
  item name, [sub-item count] type designator [sub-item length]
       [(read class list/write class list)];
```

references: IMAGE Reference Manual, Section II, Data Types
Section III, Item Part/Data Types

ITEM PART TYPE DESIGNATORS

WORD DESIGNATORS

I	A signed binary integer in 2's complement form.
J	Same as I but QUERY allows only numbers conforming to specifications for COBOL COMPUTATIONAL data to be entered.
K	An absolute binary quantity.
R	A real (floating point) number.

CHARACTER DESIGNATORS

U	An ASCII character string containing no lowercase alphabetic characters.
X	An unrestricted ASCII character string.
Z	A zone decimal format number.

NIBBLE DESIGNATOR

P	A packed decimal number.

notes:

references:

ITEM PART

TYPE DESIGNATORS AND PROGRAMMING LANGUAGES

	COBOL	FORTRAN	RPG	SPL	BASIC	LEN
I	COMPUTATIONAL S9 to S9(4)	INTEGER	Binary	INTEGER	INTEGER	2
I2	COMPUTATIONAL S9(5) to S9(9)	INTEGER*4	Binary	DOUBLE INTEGER		4
I4	COMPUTATIONAL S9(10) to S9(18)		Binary			8
J	COMPUTATIONAL S9 to S9(4)	INTEGER	Binary	INTEGER	INTEGER	2
J2	COMPUTATIONAL S9(5) to S9(9)	INTEGER*4	Binary	DOUBLE INTEGER		4
J4	COMPUTATIONAL S9(10) to S9(18)		Binary			8

notes:

- Numbers in LEN column are in bytes.

references:

ITEM PART

TYPE DESIGNATORS AND PROGRAMMING LANGUAGES

	COBOL	FORTRAN	RPG	SPL	BASIC	LEN
K1		LOGICAL		LOGICAL		2
R2		REAL		REAL	REAL	4
R4		DOUBLE PRECISION		LONG	LONG	8
U	DISPLAY PICTURE A	CHARACTER	Character	BYTE	String	1
X	DISPLAY PICTURE X	CHARACTER	Character	BYTE	String	1
Z	DISPLAY PICTURE 9		Character			1
P	COMPUTATIONAL—3		Numeric			1/2

notes:

references:

ITEM PART

```
ITEMS:
    CITY-ABBR,          X 4    (10,20/30);   << CITY ABBREVIATION          >>
    CITY-NAME,          X 20   (10,20/30);   << CITY NAME                  >>
    LIST-PRICE,         I 1    (10,20/30);   << LIST PRICE TO NEAREST $1K  >>
    CURRENT-OWNER,      X 20   (20/30);      << CURRENT OWNER               >>
    SOLD-FLAG,          X 2    (10/20,30);   << "Y" OR BLANK              >>
    SQUARE-FEET,        X 8    (10,20/30);   << SQUARE FEET               >>
```

* sub-item length denotes the length of each sub-item within an item
  - May be from 1 to 255 depending on the type designator
* Each data item length MUST be an even number of bytes

notes:

```
ITEMS:
  item name, [sub-item count] type designator [sub-item length]
       [(read class list/write class list)];
```

references:

IMAGE Reference Manual, Section III, Item Part/Data Item Length

ITEM PART

```
ITEMS:
    CITY-ABBR,          X4    (10,20/30);   << CITY ABBREVIATION          >>
    CITY-NAME,          X20   (10,20/30);   << CITY NAME                  >>
    LIST-PRICE,         I1    (10,20/30);   << LIST PRICE TO NEAREST $1K  >>
    CURRENT-OWNER,      X20   (20/30)   ;   << CURRENT OWNER              >>
    SOLD-FLAG,          X2    (10/20,30);   << "Y" OR BLANK               >>
    SQUARE-FEET,        X8    (10,20/30);   << SQUARE FEET               >>
```

* read class list/write class list specifies which user classes are permitted read and write access to the data item
  - User class numbers defined in password part
  - Write access implies read access
* More detail in security discussion

notes:

```
ITEMS:
  item name, [sub-item count] type designator [sub-item length]
       [(read class list/write class list)];
```

references: IMAGE Reference Manual, Section II, Read & Write Class Lists
Section III, Item Part

SET PART

```
SETS:

    NAME:       LIST-PRICE-MSTR, AUTOMATIC (10,20/30);
    ENTRY:      LIST-PRICE (1);
    CAPACITY:   307;

    NAME:       CITY-MASTER, MANUAL (10,20/30);
    ENTRY:      CITY-ABBR (1),
                CITY-NAME ;
    CAPACITY:   101;
```

* Defines each data set in data base
* data set name may be up to 16 alphanumeric characters
  - First character must be alphabetic

notes:

```
SETS:

                    {Manual    }
  NAME: set name,{Automatic}[(read class list/write class list)];
                    {Detail    }

  ENTRY:

  CAPACITY: maximum entry count;
```

references: IMAGE Reference Manual, Section III, Set Part

SET PART

```
SETS:

    NAME:       LIST-PRICE-MSTR, AUTOMATIC (10,20/30);
    ENTRY:      LIST-PRICE (1);
    CAPACITY:   307;

    NAME:       CITY-MASTER , MANUAL (10,20/30);
    ENTRY:      CITY-ABBR (1),
                CITY-NAME ;
    CAPACITY:   101;
```

* Denotes which type of dataset (A,M, or D)
* read/write class list specifies which user classes are permitted read and write access to the data set
  - User class numbers defined in password part
  - Write access implies read access

notes:

```
SETS:

                    {Manual   }
  NAME: set name,   {Automatic}[(read class list/write class list)];
                    {Detail   }

  ENTRY:

  CAPACITY: maximum entry count;
```

references:

SET PART

```
SETS:

    NAME:       LIST-PRICE-MSTR , AUTOMATIC (10,20/30);
    ENTRY:      LIST-PRICE (1);
    CAPACITY:   307;

    NAME:       CITY-MASTER , MANUAL (10,20/30);
    ENTRY:      CITY-ABBR (1),
                CITY-NAME;
    CAPACITY:   101;
```

* ENTRY specifies which data items belong to the set

* The order the data items appear here determines physical sequence in the entry

notes:

references:

SET PART

```
SETS:

    NAME:       LIST-PRICE-MSTR , AUTOMATIC (10,20/30);
    ENTRY:      LIST-PRICE (1);
    CAPACITY:   307 ;

    NAME:       CITY-MASTER , MANUAL (10,20/30);
    ENTRY:      CITY-ABBR (1),
                CITY-NAME ;
    CAPACITY:   101;
```

* CAPACITY specifies the maximum number of entries in a data set

* All disc space is reserved at data base creation time

notes:

references:

SET PART FOR MASTERS

```
SETS:

  NAME:       LIST-PRICE-MSTR, AUTOMATIC (10,20/30);
  ENTRY:      LIST-PRICE(1);
  CAPACITY:   307;

  NAME:       CITY-MASTER, MANUAL (10,20/30);
  ENTRY:      CITY-ABBR(1),
              CITY-NAME;
  CAPACITY:   101;
```

* path count indicates the number of paths to various detail data sets
  - May be up to 16
  - Also specifies the search item
  - If 0, the master is a stand-alone

notes:

```
                  {Manual   }
NAME: set name,   {M        }[(read class list/write class list)];
                  {Automatic}
                  {A        }

ENTRY: item name [(path count)], ...;

CAPACITY: maximum entry count;
```

references: IMAGE Reference Manual, Section II, Paths
Section III, Set Part (Masters)

SET PART FOR DETAILS

```
NAME:       RESIDENTIAL, DETAIL (10,20/30);
ENTRY:      CITY-ABBR (!CITY-MASTER),
            LIST-PRICE (LIST-PRICE-MSTR (SQUARE-FEET) ),
            CURRENT OWNER,
            SOLD-FLAG,
            SQUARE-FEET;
CAPACITY:   300;
```

* master set name indicates a path from the specified master

  —Also specifies a search item

* Up to 16 search items

* ! denotes primary path — more detail later

notes:

```
                    {Detail}
NAME: set name, {D      } [(read class list/write class list)];

ENTRY: item name [([!]master set name [(sort item name)])],...;

CAPACITY: maximum entry count;
```

references: IMAGE Reference Manual, Section II, Primary Paths
Section III, Set Part (Details)

SET PART FOR DETAILS

```
NAME:       RESIDENTIAL, DETAIL (10,20/30);
ENTRY:      CITY-ABBR (ICITY-MASTER),
            LIST-PRICE (LIST-PRICE-MSTR (SQUARE-FEET) ),
            CURRENT OWNER,
            SOLD-FLAG,
            SQUARE-FEET;
CAPACITY:   300;
```

* sort item name denotes a chain to be maintained logically in sorted sequence
  - Must be an item in the current entry
  - Must be type U,K, or X

notes:

```
                    {Detail}
NAME: set name, {D      } [(read class list/write class list)];

ENTRY: item name [([!]master set name [(sort item name)])],...;

CAPACITY: maximum entry count;
```

references: IMAGE Reference Manual, Section II, Sort Items
Section III, Set Part (Details)

END.

* Don't forget!

notes:

references:

IMAGE LIMITS

DATA ITEM NAMES PER DATA BASE	255	
DATA ITEM NAMES PER DATA ENTRY	127	
DATA SETS PER DATA BASE	99	
MAXIMUM ENTRY SIZE	4,094	BYTES
ENTRIES PER DATA SET	8,388,607	
ENTRIES PER CHAIN	65,535	
DATA ITEM LENGTH	4,094	BYTES

notes:

references:

SCHEMA

PROCESSOR

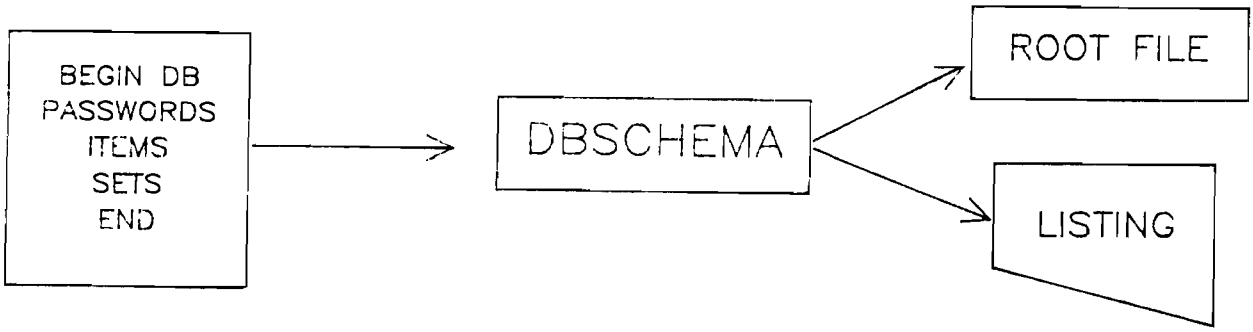
(DBSCHEMA)

notes:

references:

DBSCHEMA

* Processes SCHEMA textfile
* Produces output listing
* Creates root file



notes:

references:

ROOT FILE

* Internal description of data base design
  (machine readable form of Schema)
* Actual name is that specified in "BEGIN DATA BASE" clause of SCHEMA
* MPE commands can not access

ACCOUNT= IMAGE GROUP= PUB

FILENAME	CODE	LOGICAL RECORD					SPACE		
		SIZE	TYPE	EOF	LIMIT	R/B	SECTORS	#X	MX
REALTY	PRIV	128W	FB	3	3	1	9	1	1

HEWLETT PACKARD

notes:

references: IMAGE Reference Manual, Section II, Root File

DBSCHEMA FILES

* Formal file designator of input textfile is DBSTEXT

* Formal file designator of output listfile is DBSLIST

* By default these are $STDIN and $STDLIST

notes:

references: IMAGE Reference Manual, Section III, Operating Instructions

RUNNING DBSCHEMA

* :RUN DBSCHEMA.PUB.SYS;PARM=n

* n denotes what actual file to use for input and output

PARM=	DBSTEXT	DBSLIST
1	:FILE	$STDLIST
2	$STDIN	:FILE
3	:FILE	:FILE
OMITTED	$STDIN	$STDLIST

notes:

references:

EXAMPLE

```
:FILE DBSTEXT=LAB4SCHM
:FILE DBSLIST;DEV=LP
:RUN DBSCHEMA.PUB.SYS;PARM=3
```

* Uses textfile named LAB4SCHM
* Outputs listing to the line printer

notes:

references:

SCHEMA PROCESSOR COMMANDS

* Specify options while processing schema
* Format is: $command [parameter list]
* May be used anywhere in schema
* May be continued with & character
* $ must be in column 1

notes:

references:

IMAGE Reference Manual, Section III, Schema Processor Commands

SCHEMA PROCESSOR COMMANDS

* $CONTROL

* Parameters are:

– LIST	– NOLIST
– TABLE	– NOTABLE
– ERRORS = nnn	
– LINES = nnnnn	
– ROOT	– NOROOT
– BLOCKMAX = nnnn	

notes:

Default is

```
$CONTROL LIST, ERRORS=100, LINES=60,&
$        ROOT, TABLE, BLOCKMAX=512
```

references: IMAGE Reference Manual, Section III, $CONTROL

SCHEMA PROCESSOR COMMANDS

* $PAGE
* Parameter is "character string"
* Causes listfile to skip to top of page and print character string
* $CONTROL LIST must be in effect

check A-2

notes:

references: IMAGE Reference Manual, Section III, $PAGE

SCHEMA PROCESSOR COMMANDS

* $TITLE
* Parameter is "character string"
* Causes "character string" to be printed whenever page heading is printed
* May be overridden by $PAGE command or subsequent $TITLE command

notes:

references: IMAGE Reference Manual, Section III, $TITLE

DATA BASE

UTILITY

(DBUTIL)

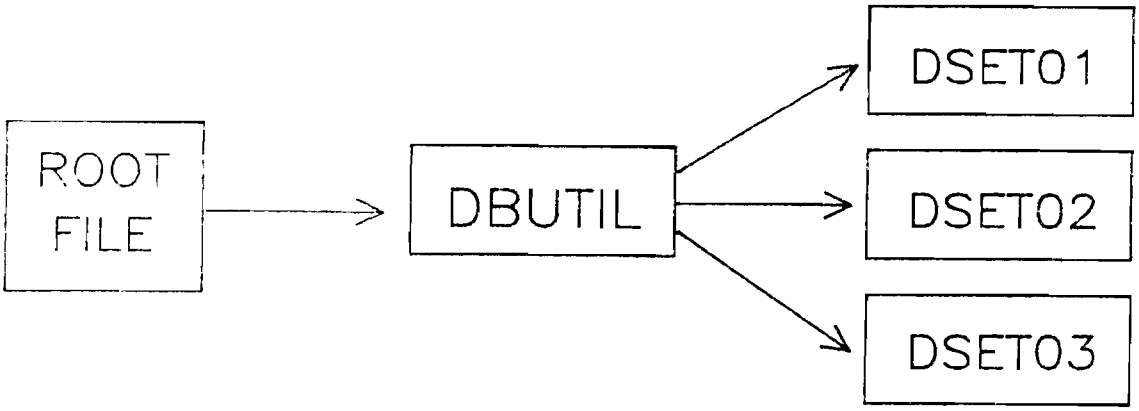
notes:

references:

DBUTIL

* Manages the data base

* Optionally creates, erases, or purges data base



notes:

references:

DBUTIL COMMANDS

:RUN DBUTIL.PUB.SYS
>>CREATE REALTY

* Allocates disc space for datasets
* Initializes datasets to binary zeros
* Saves datasets as privileged MPE files
* Names data sets:
  REALTY01, REALTY02, ... REALTYnn
* May assign maintenance word

notes:

CREATE data base name[/maintenance word]

references: IMAGE Reference Manual, Section VIII, DBUTIL CREATE

DATA BASE FILES

```
:LISTF REALTY@,2

ACCOUNT = IMAGE        GROUP = PUB

FILENAME  CODE  -------- LOGICAL RECORD-----    --- SPACE ---
                  SIZE  TYP      EOF     LIMIT R/B SECTORS #X MX
REALTY     PRIV   128W  FB         3         3   1       9  1  1
REALTY01   PRIV   128W  FB        28        28   1      30  1  1
REALTY02   PRIV   128W  FB        21        21   1      23  1  1
REALTY03   PRIV   640W  FB        13        13   1      70  1  1
```

```
>FORM SETS

DATA BASE:  REALTY

SETS:                TYPE  ITEM COUNT  CAPACITY  ENTRY COUNT  ENTRY LENGTH  BLOCKING FACTOR
LIST-PRICE-MSTER     A     1           307       0            1             11
CITY-MASTER          M     2           101       0            12            5
RESIDENTIAL          D     5           312       0            18            24
```


notes:

references:

DBUTIL COMMANDS

```
:RUN DBUTIL.PUB.SYS
>>ERASE REALTY
```

* Reinitializes datasets to binary zeros
* Datasets remain as allocated MPE files
* Commonly used before reloading data back into data base

notes:

ERASE data base name[/maintenance word]

references: IMAGE Reference Manual, Section VIII, DBUTIL ERASE

DBUTIL COMMANDS

```
:RUN DBUTIL.PUB.SYS

>>HELP
      .
      .
      .
>>EXIT
```


notes:

- may also :RUN DBUTIL.PUB.SYS,{CREATE}
  {ERASE }
  {PURGE }
- will be prompted for data base name

references:

LAB 4

DATA BASE CREATION

notes:

references:

DATA SET NAME	ITEM NAME	#	TYPE/DESIG	KEY	PATH CT	CAPACITY	BLOCKMAX
LISTNR-MASTER			A		1	503	128
	LISTING-NR	1	I1	X			
ZONING-MASTER			M		1	31	128
	ZONING-CODE	2	X4	X			
	ZONING-DESCRIP	3	X20				
COMMERCIAL			D		2	300	768
	LISTING-NR	1	I1	X			
	ZONING-CODE	2	X4	X			
	LIST-PRICE	4	I1				
	STREET-ADDRESS	5	X30				
	SOLD-FLAG	6	X2				
	SQUARE-FEET	7	X8				

DBSCHEMA SUMMARY TABLE

DATA SET NAME	TYPE	FLD CNT	PT CT	ENTR LGTH	MED REC	CAPACITY	BLK FAC	BLK LGTH	DISC SPACE
LISTNR–MASTER	A	1	1	1 +10	11	503	11	122	48
ZONING–MASTER	M	2	1	12 +10	22	31	5	111	9
COMMERCIAL	D	6	2	24 +8	32	322	23	738	90

TOTAL DISC SECTORS INCLUDING ROOT: 156

notes:

references: IMAGE Reference Manual, Section III, Summary Information

DATA BASE

ACCESS

notes:

references:

TOPICS

* DATA BASE PROCEDURES

* MULTIPLE USER CONSIDERATIONS
  - Concurrent Access
  - Open Modes
  - IMAGE Locking

* SECURITY

notes:

references:

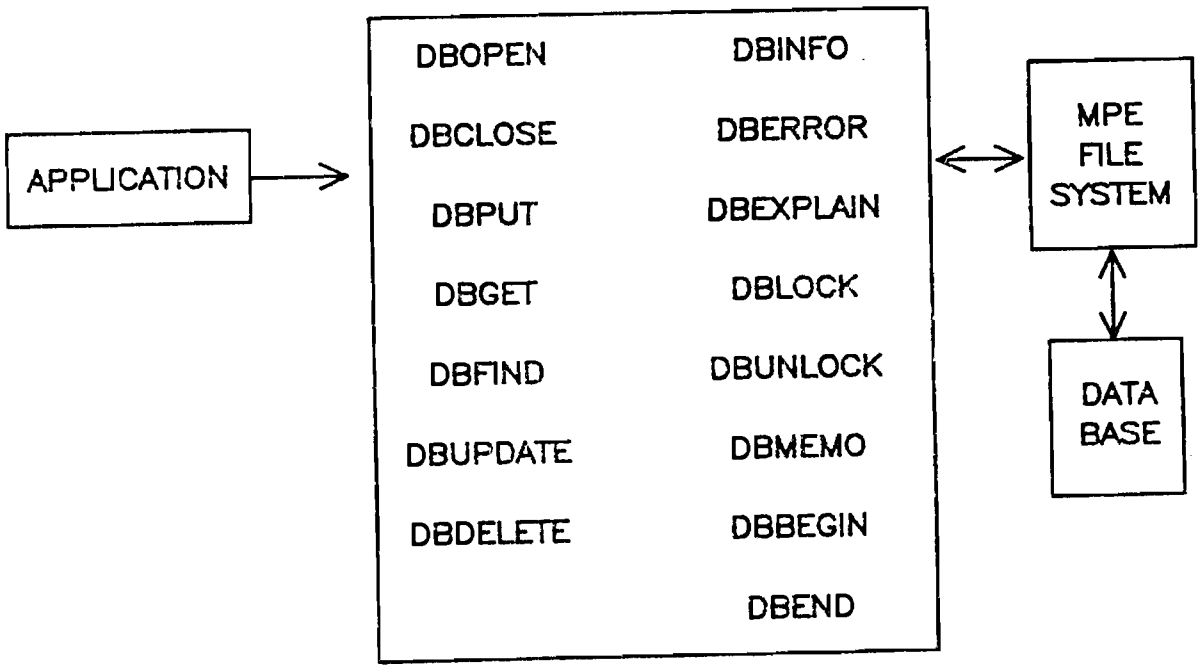
DATA BASE

PROCEDURES

notes:

references:

IMAGE PROCEDURES



notes:

references:

IMAGE PROCEDURES

SPL

* Self contained sections of code invoked by a "CALL" statement in an application program

* Can be passed parameters

* Can return values

* External to the application program

SL.PUB.SYS

notes:

references:

CALLING AN IMAGE PROCEDURE

* COBOL CALL "name" USING parm,parm,...
* BASIC CALL name (parm,parm,...)
* SPL name (parm,parm,...)
* FORTRAN CALL name (parm,parm,...)
* PASCAL name (parm,parm,...)

notes:

- RPG accesses IMAGE procedures indirectly

references:

PARAMETERS

* All parameters are required

* Data types vary

* Must be passed by reference

* Must begin on word boundaries

COBOL variables are aligned on word boundaries if a 77 or 01 level is used.

means that when program sees the parameter in the data stack is not starting on the second byte of a word rather on the next one.

notes:

references:

PARAMETER DATA TYPES

* ARRAY

```
01 DSET-NAME PIC X(16) VALUE "CITY-MASTER;".
```

* INTEGER

```
01 MODE PIC 9999 COMP VALUE 1.
```

* DOUBLE INTEGER

```
01 REC-NUM PIC S9(9) COMP.
```

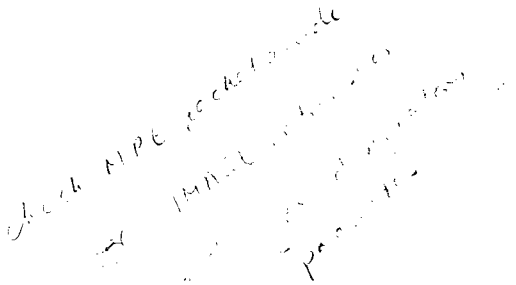
* INTEGER ARRAY

```
01 STATUS
   05 COND-WORD PIC S9(4) COMP.
   05 STAT1     PIC S9(4) COMP.
   05 STAT2-3   PIC S9(9) COMP.
      :
```


notes:

- types denoted as:

Array	A
Integer	I
Double Integer	D
Integer Array	IA



references: See APPENDIX C for specific language examples
IMAGE Reference Manual, Section VI, Language Considerations and Examples

PASSING PARAMETERS

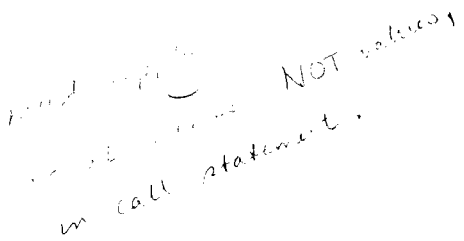
* By reference, the address of the parameter is passed to the procedure

  Ex:

  CALL DBCLOSE (BASE, DSET, MODE, STATUS)

* Exception: BASIC may pass by value

notes:



references:

IMAGE PARAMETERS

* Status array is a parameter included in every IMAGE procedure call

* Ten word array

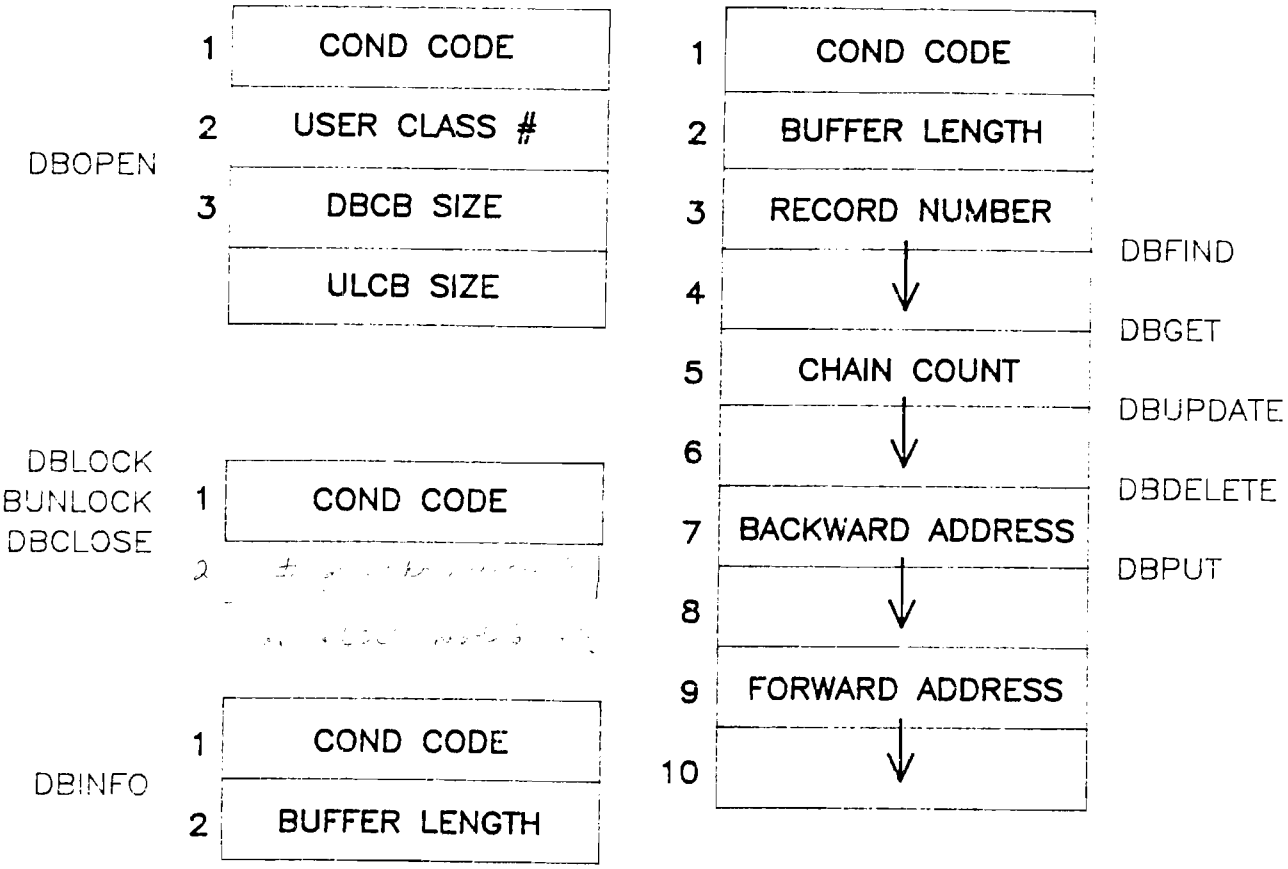
* Returns the condition code in the first word

* The values returned in the other words vary with the call

notes:

references: IMAGE Reference Manual, Section IV, Checking the Status of a Procedure

STATUS PARAMETER



notes:

references: IMAGE Reference Manual, Section V

STATUS PARAMETER

* The condition code is represented numerically
  - 0 means successful
  - <0 means procedure failed
  - >0 means exception other than failure
* Specific exceptions vary with each call

notes:

references:

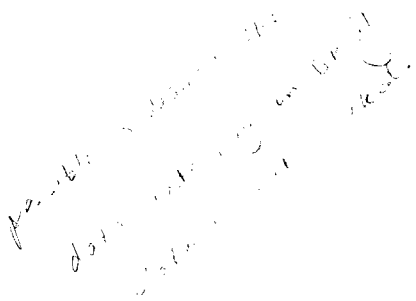
IMAGE PARAMETERS

ALWAYS CHECK

THE STATUS

AFTER EACH CALL

notes:



references:

DBOPEN
(base, password, mode, status)

* Initiates user access to data base

* Opens root file only —

notes:

references: IMAGE Reference Manual, Section V, DBOPEN
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DBOPEN

Base parameter

- Identifies the data base being accessed
- Characters 1 and 2 must be left blank
- This parameter must not be modified or redefined once used in the DBOPEN
- Must be terminated by ";" or blank

notes:

```
          A        A        I        IA
DBOPEN (base, password, mode, status)
```

references:

DBOPEN

Password parameter

- Values defined in SCHEMA
- Determines user class number
- If <8 characters must be terminated by ";" or blank
- ";" is a valid password for creator only
- ";" gives unlimited access

notes:

```
          A       A          I        IA
DBOPEN (base, password, mode, status)
```

references:

DBOPEN

Mode parameter

- Determines a user's access to data base
- May or may not allow concurrent access
- Mode 3 is exclusive modify access
- Mode 7 is exclusive read access
- More on modes under multiple user considerations

notes:

```
                A         A         I         IA
DBOPEN (base, password, mode, status)
```

references:

```
    7           15    19      26     30                    53         60
F   CITY        UC    F       24R    4AM
F                                                          KIMAGE  REALTY3
F                                                          KLEVEL  MANAGER
F                                                          KSTATUSSTAT
```

* KIMAGE spec defines the data base name and mode
* KLEVEL spec defines the password
* KSTATUS spec defines a 6 element status array

notes:

references: IMAGE Reference Manual, Section VI, RPG Examples
RPG Reference Manual, Section IV, Data Base Management
Group Fields

BASIC

* BIMAGE procedures simplify access to data base

* Format same as IMAGE procedures but name preceded by an X

* Certain parameters may be passed by value

notes:

```
XDBOPEN (base, password, mode, status)
```

references: IMAGE Reference Manual, Section VI, Basic Examples

DBPUT
(base, dset, mode, status, list, buffer)

* Adds a new entry to a data set
* Detail and manual master data sets only
* If detail add:
  - Manual master must first contain same search item value
  - Automatic master automatically maintained

notes:

references: IMAGE Reference Manual, Section V, DBPUT
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DBPUT

Dataset parameter

- Identifies the dataset being accessed
- May be an integer referencing the dataset number
- 16 characters maximum if name

Mode parameter

- Always a 1

notes:

```
          A     IorA   I       IA       A       A
DBPUT (base, dset, mode, status, list, buffer)
```

references:

DBPUT

Status parameter

- Common exceptional condition codes for detail

  1nn Missing master entry for path number nn

for master

43 Duplicate search item

notes:

```
          A     IorA   I       IA      A       A
DBPUT (base, dset, mode, status, list, buffer)
```

references:

DBPUT

List parameter

- Specifies the data items to be given values for the new entry
- Names or numbers
- Any item not specified is binary zero filled
- "@;" specifies all items
- "*;" specifies the previous list used for this dataset

notes:

```
        A     IorA   I       IA      A       A
DBPUT (base, dset, mode, status, list, buffer)
```

references:

DBPUT

Buffer parameter

- Specifies the area containing the data to be added
- Order and type of variables must correspond to list parameter

check length of buffer is correct in last parameter

notes:

```
          A     IorA   I       IA      A       A
DBPUT (base, dset, mode, status, list, buffer)
```

references:

BUFFER PARAMETER FOR BASIC

* BIMAGE procedures automatically pack list of basic variables or expressions into buffer

notes:

references: IMAGE Reference Manual, Section VI, Basic Examples

RPG

```
      7        15    19       26     30                   53      60     66
F     CITYA    UC    F        24R    4AM                                 A
F                                                         KIMAGE  REALTY3
F                                                         KLEVEL  MANAGER
F                                                         KSTATUSSTAT
                                                          KDSNAMECITY-MASTER
```

* File addition constructs used

* KDSNAME spec defines the data set name

* No partial lists supported

notes:

references: RPG Reference Manual, Section IV, File Sharing Group Fields

DBINFO
(base, qualifier, mode, status, buffer)

* Returns information about data base structure

* Modes for data item, data set, path, and logging info

* Accesses the root file

* Commonly used to retrieve data item and data set numbers for subsequent use in IMAGE calls

notes:

references: IMAGE Reference Manual, Section V, DBINFO
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DBEXPLAIN (status)

* Interprets status array

* Outputs multi—line message to $STDLIST

* Sample output:

```
IMAGE RESULT AT %001057:CONDITION WORD = 16
DBPUT, MODE 1, ON #1 OF COMPNY
THE DATA SET IS FULL
```

Sample output:

```
IMAGE RESULT: CONDITION WORD = 5349
IMAGE CALL INFORMATION NOT AVAILABLE
UNRECOGNIZED CONDITION WORD: 5349
OCTAL DUMP OF STATUS ARRAY FOLLOWS:
012345 054321 011111 022222 033333 044444 055555
066666 077777 000000
```


notes:

references: IMAGE Reference Manual, Section V, DBEXPLAIN
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DBERROR
(status, buffer, length)

* Interprets status array
* Returns up to 72 ASCII characters of an english language message
* Buffer parameter:
  Area where message is returned
* Length parameter:
  Variable where byte length of message is returned
* Sample output:
  DUPLICATE KEY VALUE IN MASTER

notes:

references: IMAGE Reference Manual, Section V, DBERROR
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DBCLOSE
(base, dset, mode, status)

* Mode 1 closes data base
* Mode 2 closes data set
* Mode 3 "rewinds" data set — pointers reset to start.
* Dataset parameter ignored for mode 1, but still required

HEWLETT PACKARD

notes:

references: IMAGE Reference Manual, Section V, DBCLOSE
MPE Pocket Guide, Section IV, IMAGE Intrinsics

LAB 5

BASIC DATA BASE ACCESS

(OPEN,CLOSE,ADD DATA,

CHECK ERRORS)

notes:

references:

LAB 5 -- BASIC DATA BASE ACCESS (OPEN, CLOSE, ADD DATA, CHECK ERROR

1. INTRODUCTION

A data base is of no use to anyone unless it contains data. In this lab you will modify a skeleton program to add data into the REALTY data base. Many real applications use HP's VPLUS as a terminal screen handler. This system permits you to design your own screen layouts to facilitate input and output of data to the terminal. The required subroutine calls as well as the form file have been coded for you. Under user control, you will be able to load either city codes into the master data set or individual property records into into the residential detail data set.

2. OBJECTIVES

(1) Gain familiarity with the following basic IMAGE calls:
DBOPEN, DBPUT, DBEXPLAIN and DBCLOSE.

(2) Understand the necessity and technique of checking the status parameter after EVERY IMAGE procedure call.

3. INSTRUCTIONS

A. FILES:

LAB5xxxS.LABS	Skeleton source program for LAB 5 (where xxx = BAS, COB, FTN, PAS, RPG, SPL)
FASTFORM.LABS	VPLUS fast-compiled form file.
REALTY.groupn	Your data base
LAB5xxxP.groupn	Compiled/prep'd program file.

B. DETAILED INSTRUCTIONS:

1. Text LAB5xxxS.LABS into the EDITOR. Add code as required to open the data base for exclusive access.
2. Add code to write data to the CITY-MASTER and RESIDENTIAL detail data sets.
3. Add code to close the data base.
4. Add code to process IMAGE procedure call errors, and to CHECK STATUS after EVERY procedure call.
   Note: No errors are acceptable for DBOPEN and DBCLOSE. Non-fatal errors allowed (check for these before you check for zero) are:
   DBPUT: 43, 1xx.
   These are errors which should result in an error message and a retry of corrected input. All other non-zero errors are fatal and should result in programatic abort following calls to DBEXPLAIN.
5. Save the source program in your group as LAB5xxxS.
6. Compile LAB5xxxS.
7. When you have a clean compile, prep your program:
   PREP $OLDPASS,LAB5xxxP;MAXDATA=20000
8. Run LAB5xxxP. (remember xxx = language code!)

C. SOLUTION:
The solution of this lab can be found in LAB6xxxS.LABS.
Please refer to this AFTER you have attempted the lab on your own.

DBFIND
(base, dset, mode, status, item, argument)

* Locates the detail chain head in the related master for subsequent access to a detail chain

* Must be called before any chained access to detail data set

notes:

references: IMAGE Reference Manual, Section V, DBFIND
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DBFIND

Dataset parameter

- Identifies the detail data set to be accessed

Mode parameter

- Always a 1

Status parameter

- Common exceptional condition codes

17 No master entry — can be used to link end of chain

notes:

```
          A     IorA   I       IA      IorA      A
DBFIND (base, dset, mode, status, item, argument)
```

references:

DBFIND

Item parameter

- Identifies the search item name or number
- 16 characters maximum if name

Argument parameter

- Identifies the search item value

notes:

```
                A     IorA   I       IA     IorA      A
DBFIND (base, dset, mode, status, item, argument)
```

references:

DBGET
(base, dset, mode, status, list, buffer, argument)

* Reads the data items of a specified entry
* 8 different modes of access
* 4 different methods of access

notes:

references: IMAGE Reference Manual, Section V, DBGET
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DATA BASE ACCESS METHODS

* SERIAL
* DIRECTED
* CALCULATED
* CHAINED

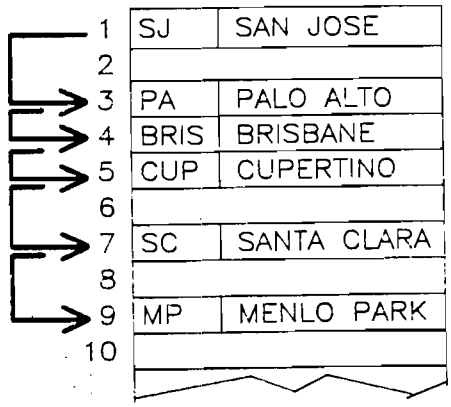
notes:

references:

SERIAL ACCESS

* Access begins at first or current entry
* Only "active" entries are read
* Backward or forward access

DBGET
- Mode 2
- Mode 3



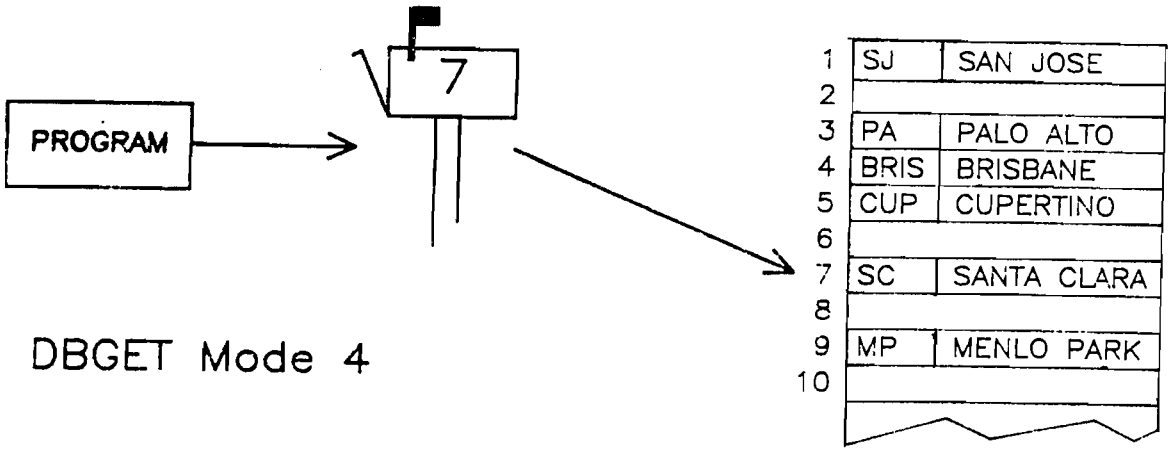
notes:

references: IMAGE Reference Manual, Section IV, Serial Access

DIRECTED ACCESS

* Program specifies the relative record address

* If active entry at specified address, data is returned



DBGET Mode 4

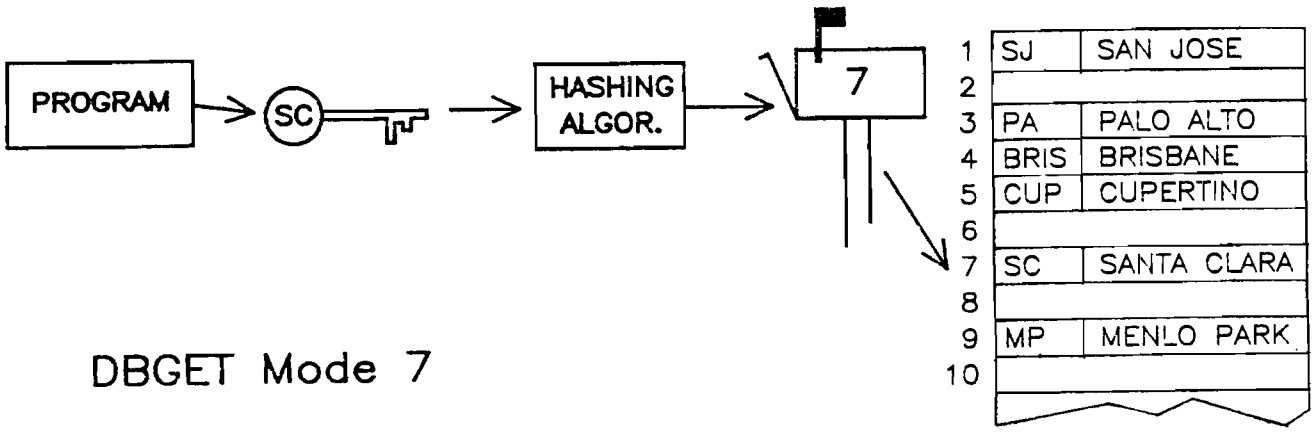
notes:

references: IMAGE Reference Manual, Section IV, Directed Access

CALCULATED ACCESS

* Program specifies search item value

* Master entry access only



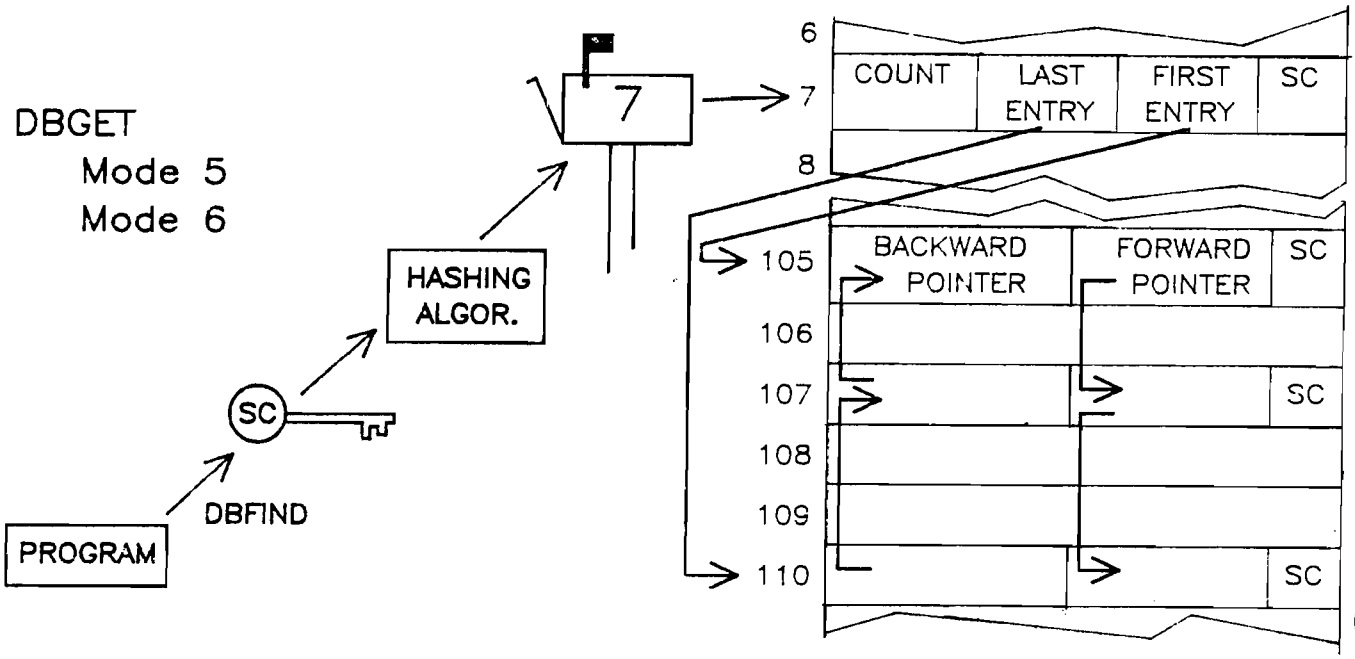
DBGET Mode 7

notes:

references: IMAGE Reference Manual, Section IV, Calculated Access

CHAINED ACCESS

* Access begins at first or last entry in detail chain
* Backward or forward access
* Must "FIND" chain head in master first



notes:

references: IMAGE Reference Manual, Section IV, Chained Access

DBGET

Mode parameter

1 Re—read
2 Serial — Forward
3 Serial — Backward
4 Directed
5 Chained — Forward
6 Chained — Backward
7 Calculated
8 Primary calculated

notes:

```
                A     IorA   I       IA      A       A      A or DI
DBGET (base, dset, mode, status, list, buffer, argument)
```

references:

DBGET

Status parameter

– Common exceptional condition codes

for detail

15 End of chain (mode 5)

14 Beginning of chain (mode 6)

for master

17 No entry (modes 1, 4, 7, 8)

notes:

```
          A     IorA   I       IA      A       A      A or DI
DBGET (base, dset, mode, status, list, buffer, argument)
```

references:

DBGET

List parameter

- Specifies the data items whose values are to be returned
- Names or numbers
- "@;"
- "*;"
- Blank list: " ", " ;", or 0

used instead of word as print

HEWLETT PACKARD

notes:

```
          A     IorA   I       IA      A       A      A or DI
DBGET (base, dset, mode, status, list, buffer, argument)
```

references:

DBGET

Buffer parameter

- Specifies the area the data is to be returned to
- Order and type of variables must correspond to list parameter

notes:

```
          A     IorA   I       IA      A       A       A or DI
DBGET (base, dset, mode, status, list, buffer, argument)
```

references:

DBGET

Argument parameter

- Identifies search item value if mode is 7 or 8 (calculated)
- Identifies relative record address if mode is 4 (directed)
- Ignored, but required if mode is 1, 2, 3, 5 or 6

notes:

```
          A     IorA   I       IA       A       A       A or DI
DBGET (base, dset, mode, status, list, buffer, argument)
```

references:

RPG

```
     7        15   19        26   30                  53        60        67
F  RES       IC   F         90R  4AM
F                                                KIMAGE   WONDER3C
F                                                KDSNAMERESIDENTIAL
F                                                KITEM    CITY-ABBR
```

* KIMAGE spec defines access mode

C 'chained forward'

* KITEM spec defines search item name

* No partial lists supported

* Search item value or relative record address supplied in CHAIN calculation spec

notes:

references: IMAGE Reference Manual, Section VI, RPG Examples
RPG Reference Manual, Section IV, Database Management
Group Fields

RPG AND ACCESS MODES

* K spec column 67
* Mode 1 (re—read) not supported
* Mode 5 and 6 (chained reads) cause DBFIND to master, then <u>one</u> read to detail
* Use mode C or R (chained sequential reads) to read down detail chain
* Blank mode used for OUTPUT ADD

notes:

references: RPG Reference Manual, Section IV, Input/Output Modes

RPG

* Multiple access modes to same data set require multiple file descriptions

* Use same KDSNAME spec

* Use dummy data base name in KIMAGE spec and file equate to actual DB name

notes:

references: RPG Reference Manual, Section IV, File Sharing Group Fields

LAB 6

DATA RETRIEVAL

notes:

references:

LAB 6 -- DATA RETRIEVAL

1. INTRODUCTION

You have now successfully added data to the data base. This lab will build on the previous program and teach you how to find and retrieve data from various data sets.

2. OBJECTIVES

(1) Gain familiarity with the use of the following IMAGE retrieval procedures: DBFIND, DBGET.

3. INSTRUCTIONS

A. FILES:

LAB6xxxS.LABS	Skeleton source program for LAB 6 (where xxx = BAS, COB, FTN, PAS, RPG, SPL)
FASTFORM.LABS	VPLUS fast-compiled form file
REALTY.groupn	Your data base
LAB6xxxP.groupn	Compiled/prep'd program file

B. DETAILED INSTRUCTIONS:

1. Text LAB6xxxS.LABS into the EDITOR.

2. Add code to retrieve data from the CITY-MASTER data set.

3. Add code to find and retrieve data from the RESIDENTIAL detail data set using CITY-ABBR as the search key.

4. CHECK STATUS after EVERY procedure call.
   Allowable (non-fatal) errors are:
   DBFIND: 17.
   DBGET : Mode 5 chained read: 15.
   Mode 7 calculated read: 17.
   All other non-zero errors are fatal.

5. Save the source program in your group as LAB6xxxS.

6. Compile LAB6xxxS.

7. When you have a clean compile, prep your program:
   PREP $OLDPASS,LAB6xxxP;MAXDATA=20000

8. Run LAB6xxxP.

C. SOLUTION:
The solution to this lab can be found in LAB7xxxS.LABS.
Please refer to this AFTER you have attempted the lab on your own.

DBUPDATE
(base, dset, mode, status, list, buffer)

* Changes values of items in the specified entry
* Must be preceded by a DBGET
* Search and sort item values cannot be changed

notes:

references: IMAGE Reference Manual, Section V, DBUPDATE
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DBUPDATE

Mode parameter

- Always a 1

Status parameter

- Common exceptional condition codes

17	No entry
41	Critical item

notes:

```
                 A     IorA   I        A        A        A
DBUPDATE (base, dset, mode, status, list, buffer)
```

references:

DBUPDATE

List parameter

- Specifies the data items to be changed
- Names or numbers
- "@;"
- "*;"

notes:

```
                A     IorA    I       A       A       A
DBUPDATE (base, dset, mode, status, list, buffer)
```

references:

DBUPDATE

Buffer parameter

- Specifies the area containing the changed data item values
- Order and type of variables must correspond to list parameter

notes:

```
                A     IorA   I        A        A        A
DBUPDATE (base, dset, mode, status, list, buffer)
```

references:

DBDELETE
(base, dset, mode, status)

* Deletes an entry of a data set
* Must be preceded by a DBGET
* Detail and manual master data sets only

notes:

references: IMAGE Reference Manual, Section V, DBDELETE
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DBDELETE

* If master delete
  - all chain heads must be empty
* If detail delete
  - cannot "delete chain"
* Automatic master entry automatically deleted when all detail chains are deleted

notes:

```
                A     IorA   I       A
DBDELETE (base, dset, mode, status)
```

references:

DBDELETE

Mode parameter

- Always a 1

Status parameter

- Common exceptional condition codes

17	No entry
44	Chain head not empty

notes:

```
                A     IorA   I       A
DBDELETE (base, dset, mode, status)
```

references:

LAB 7

DATA MODIFICATION

notes:

references:

1. INTRODUCTION

In labs 5 and 6 we learned the skills required to load and retrieve data in a data base. In this lab we will learn the mechanics of updating and deleting records from the data base. At the completion of this lab, you will have a complete,fundamental online data entry and retrieval program for use with the REALTY data base.

2. OBJECTIVES

(1) Gain familiarity with the use of data modification procedures: DBUPDATE and DBDELETE.

3. INSTRUCTIONS

A. FILES:

LAB7xxxS.LABS	Skeleton source program for LAB 7 (where xxx = BAS, COB, FTN, PAS, RPG, SPL)
FASTFORM.PUB	VPLUS fast-compiled form file
REALTY.groupn	Your data base
LAB7xxxP.groupn	Compiled/prep'd program file

B. DETAILED INSTRUCTIONS:

1. Text LAB7xxxS.labs into the EDITOR.

2. Add code to update and delete entries in the CITY-MASTER data set. The previous DB call must have been a DBGET!!!

3. Add code to update and delete entries in the RESIDENTIAL data set. You must find the first entry, then read down the detail chain to locate the proper record.

4. CHECK STATUS after EVERY procedure call.
   Allowable (non-fatal) errors are:
   DBDELETE: 17, 44.
   DBUPDATE: 17, 41, 42.
   All other non-zero errors are fatal.

5. Save the source program in your group as LAB7xxxS.

6. Compile LAB7xxxS.

7. When you have a clean compile, prep your program:
   PREP $OLDPASS,LAB7xxxP;MAXDATA=20000

8. Run LAB7xxxP.

C. SOLUTION:

The solution to this lab can be found in LAB8xxxS.LABS.
Please refer to this AFTER you have attempted the lab on your own.

MULTIPLE USER

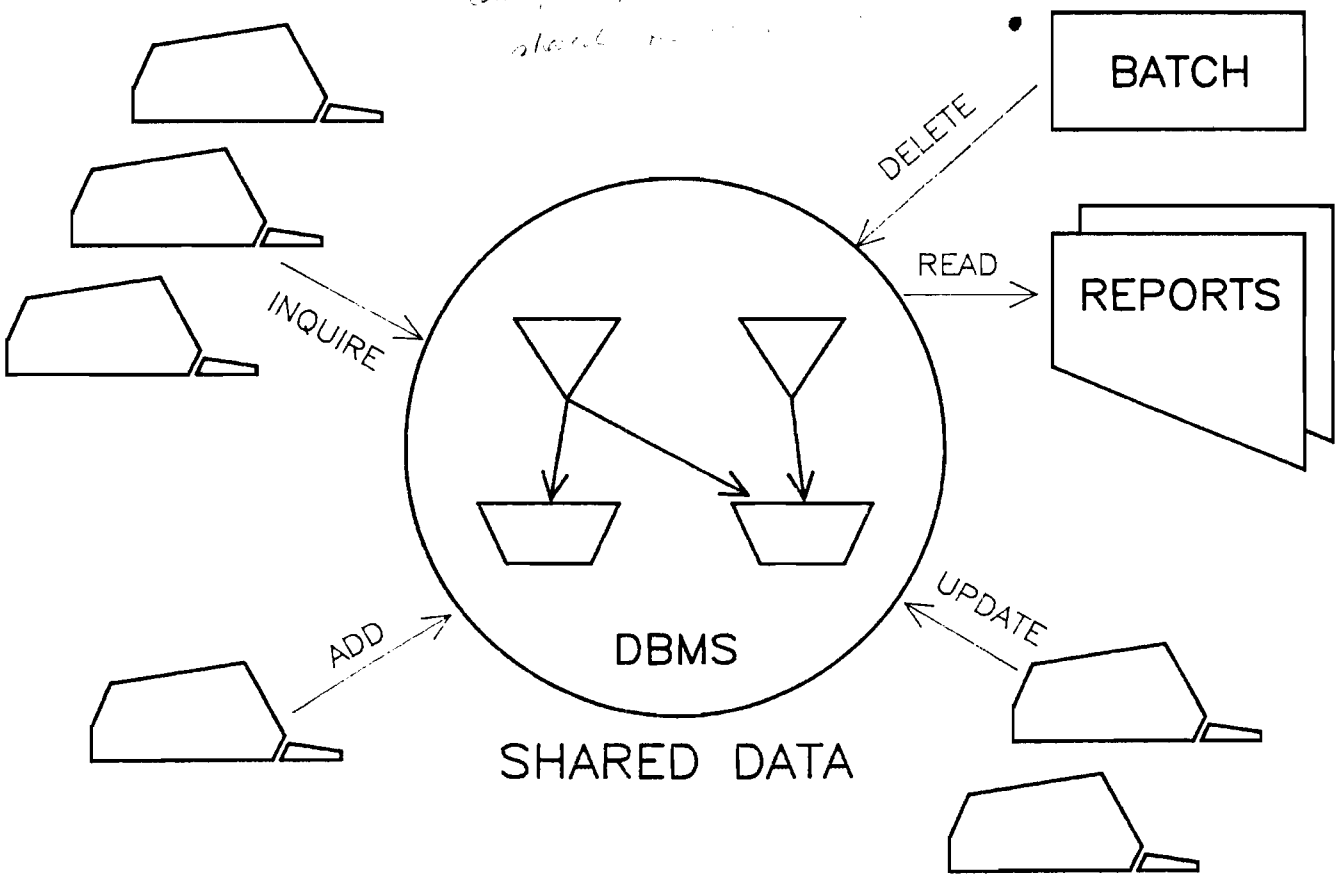
CONSIDERATIONS

* Concurrent Access
* Open Modes
* IMAGE Locking

notes:

references:

CONCURRENT ACCESS



notes:

references:

CONCURRENT ACCESS

Controlled by DBOPEN mode

* Determines access type
  - Modify, update, read-only
  - Protects data base integrity
* Coordinates shared access
  - Shared or exclusive
  - If shared, with what access types

notes:

references:

DBOPEN ACCESS TYPES

* Read = DBFIND and DBGET
* Update = read and DBUPDATE
* Modify = update and
  DBPUT and DBDELETE

notes:

references:

COMPARING UPDATE AND MODIFY ACCESS

* UPDATE allows item values to be changed
  - In place
  - Applies to data items only
  - Not search or sort items

* MODIFY allows entries to be added and deleted
  - Alters data set structure
  - Applies to data entries only
  - Add must include search and sort items

notes:

references:

8 DBOPEN MODES

* Two are exclusive

* Six allow concurrent access
  - Paired together
  - Different levels of access plus concurrent read–only

notes:

references: IMAGE Reference Manual, Section IV, Access Modes

DBOPEN MODE	ACCESS TYPE	CONCURRENT MODES ALLOWED		
		modify	update	read–only
1	modify	1		5
2	update		2	6
3	modify	none	none	none
4	modify			6

These modes allow data to be changed

notes:

references:

DBOPEN MODE	ACCESS TYPE	CONCURRENT MODES ALLOWED		
		modify	update	read-only
5	read-only	1		5
6	"	4 or 2 or		6 plus 8
7	"	none	none	none
8	"			6 plus 8

These modes are read-only

notes:

references:

Problems can arise...

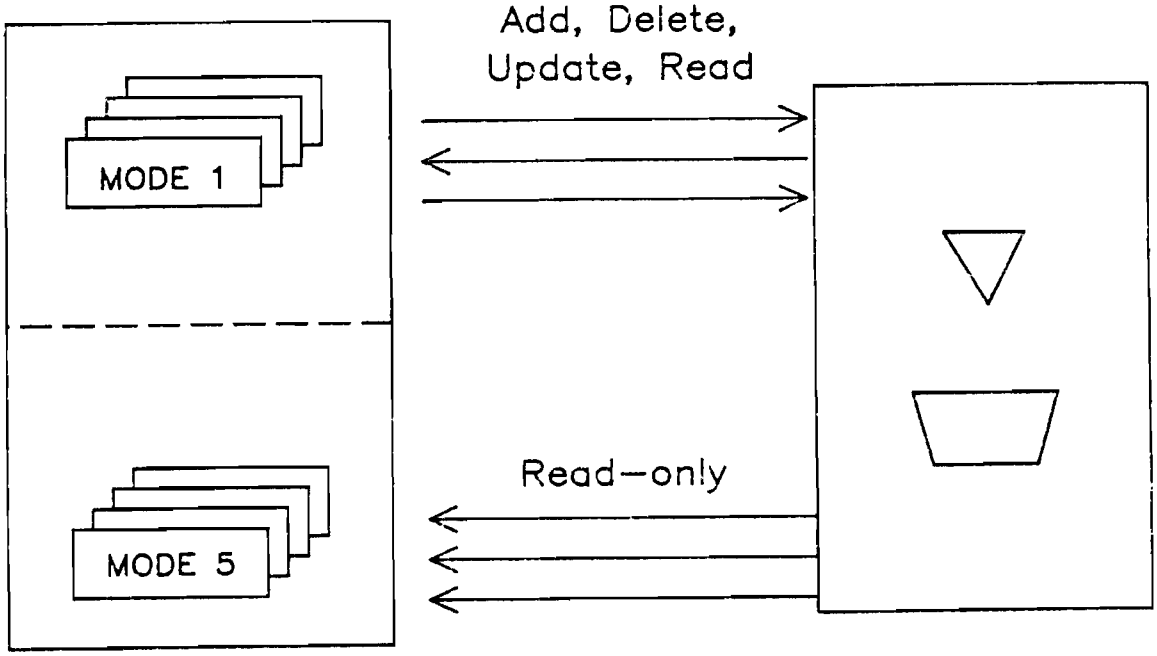
when DBOPEN modes conflict

* Plan shared DBOPEN modes during design phase
* Decision is based on type of concurrent activity required

notes:

references:

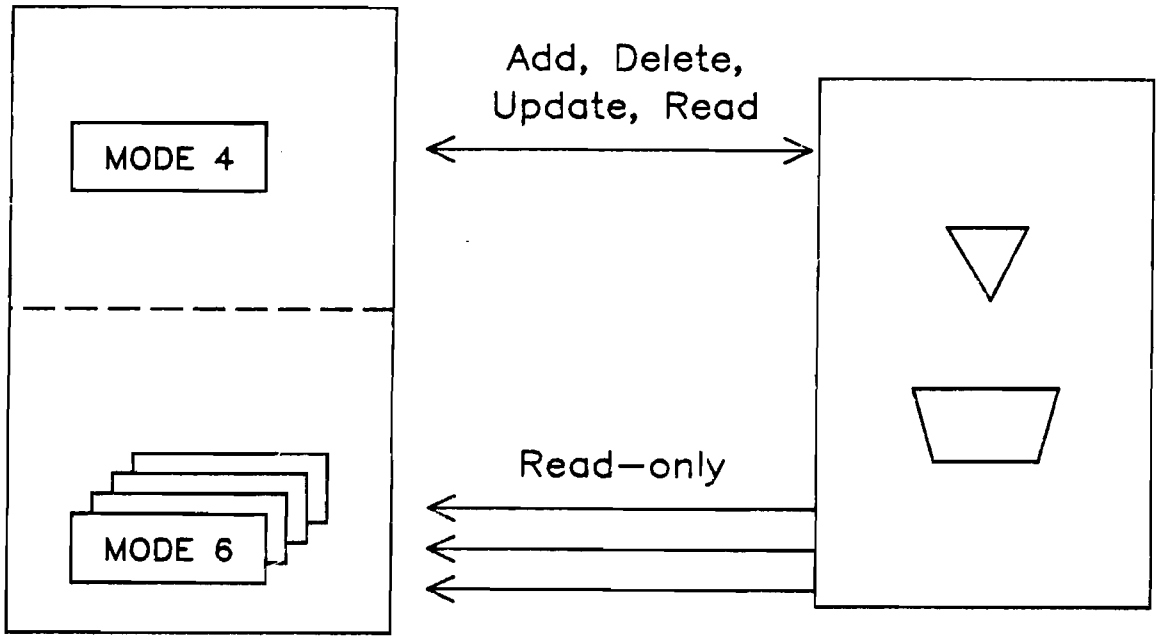
Will data base need concurrent and multiple modifiers?...



notes:

references:

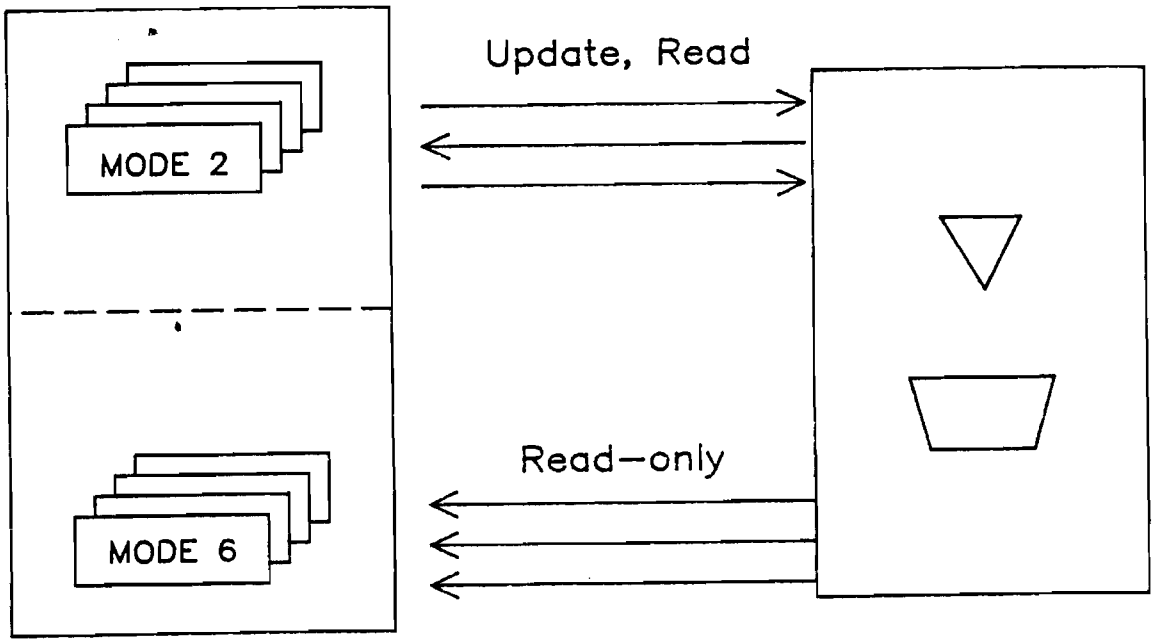
...or only a single concurrent modifier?



notes:

references:

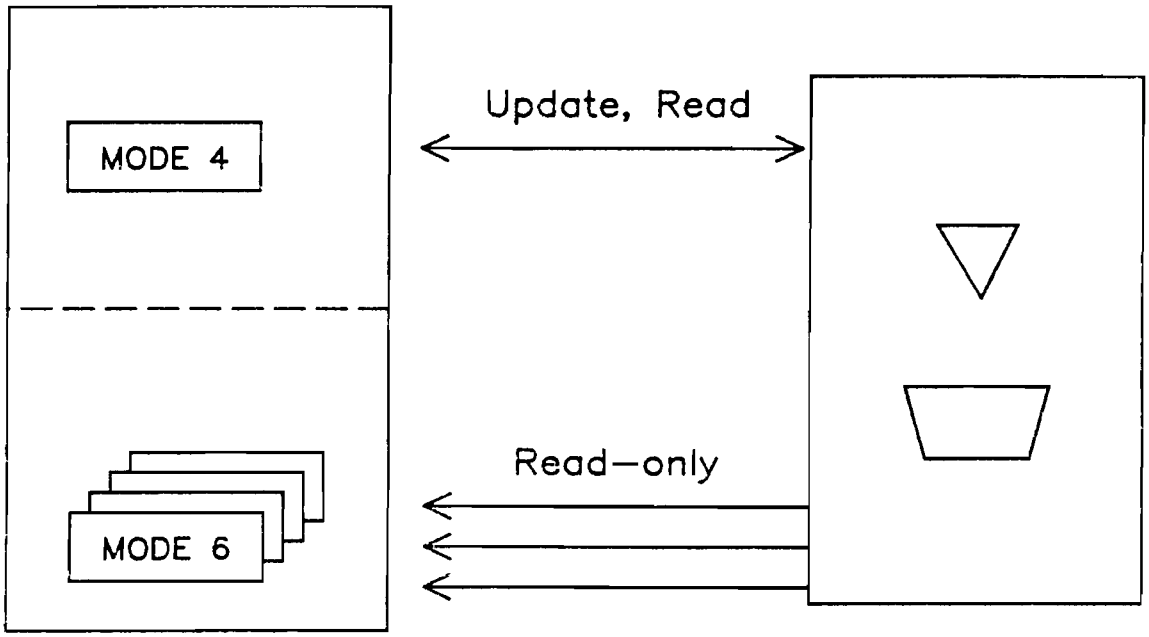
...or no modifiers, but multiple updaters?



notes:

references:

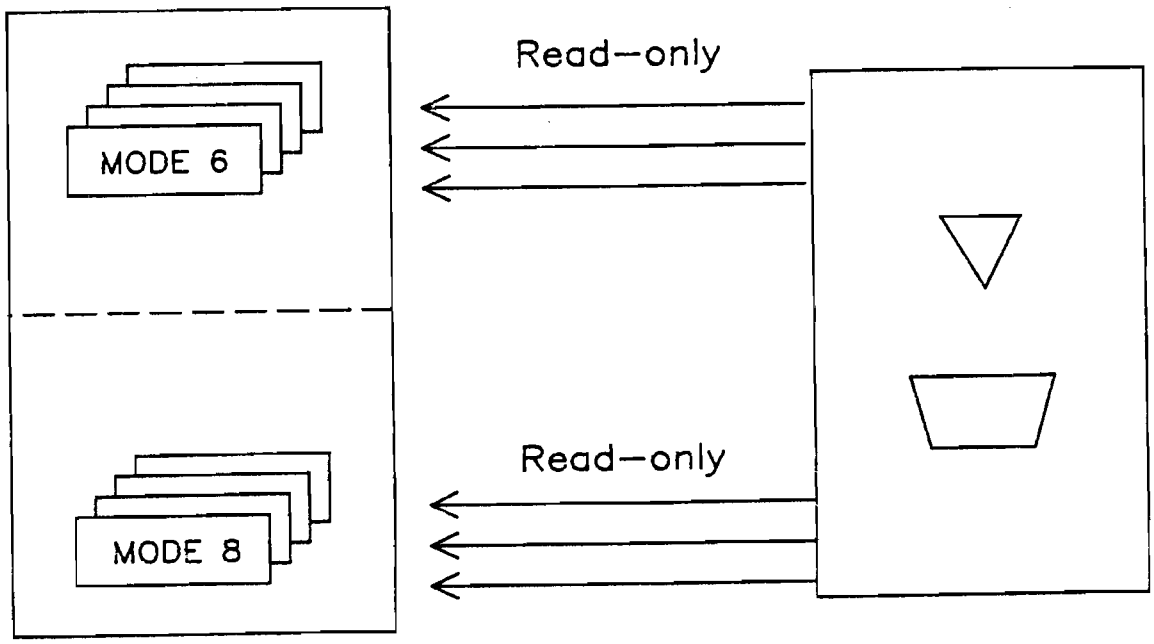
...or only 1 updater?



notes:

references:

...or simply restrict concurrent use to read—only?



notes:

references:

BUT...

with concurrent users
there needs to be a way to

COORDINATE ACCESS...

HEWLETT PACKARD

notes:

references:

LOCKING

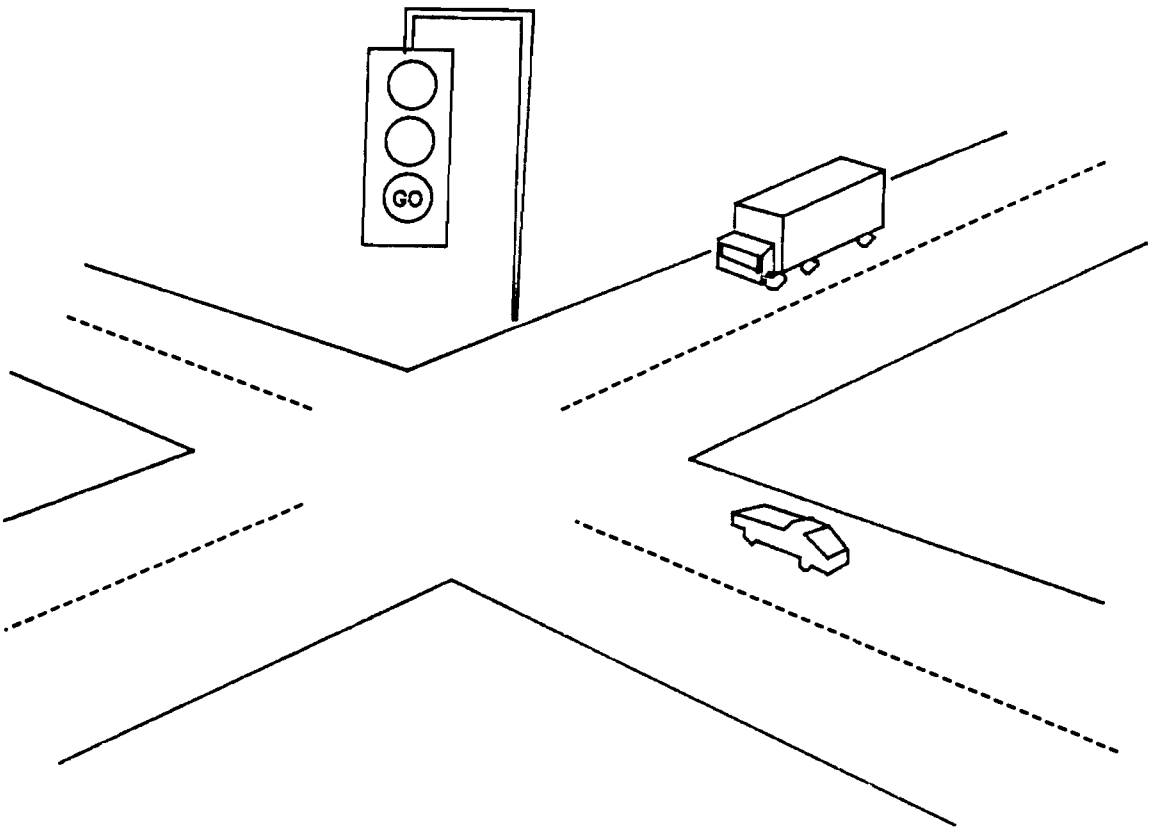
A logical (not physical) signaling facility

to control access to an IMAGE data base.

[SOFT way]

notes:

references: IMAGE Reference Manual, Section IV, Using The Locking Facility

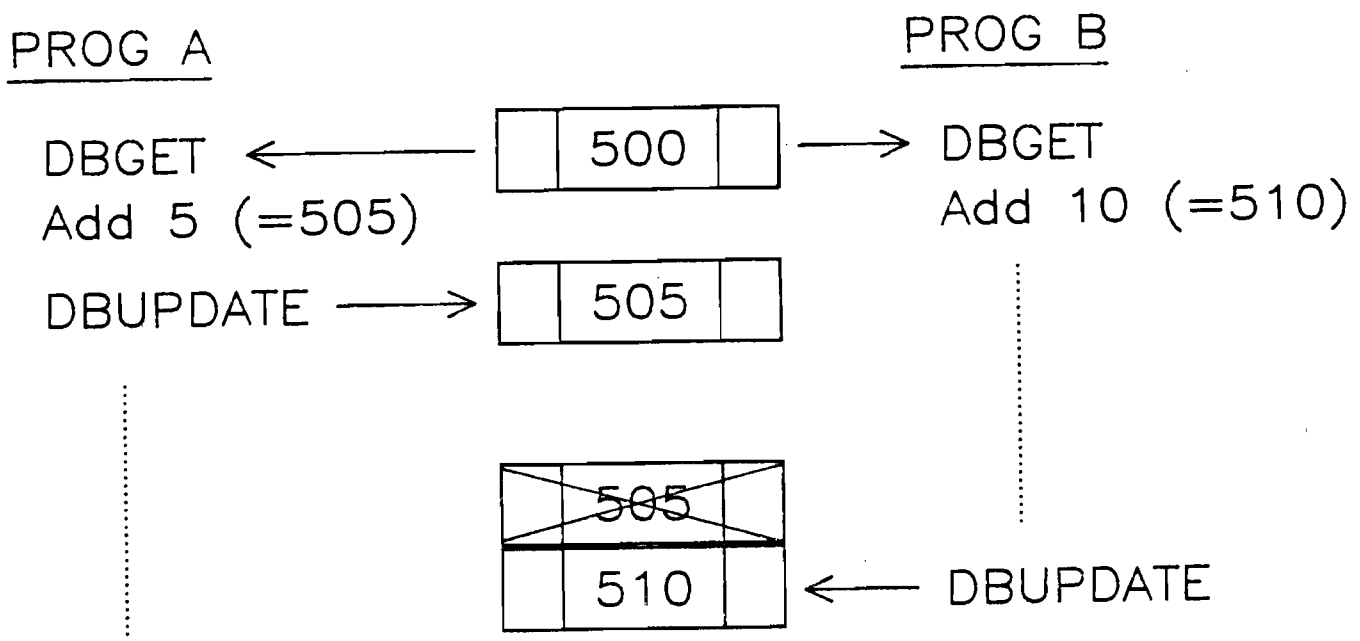


notes:

- Intersection with traffic light -- requires COOPERATION

references:

UPDATE WITHOUT LOCKING

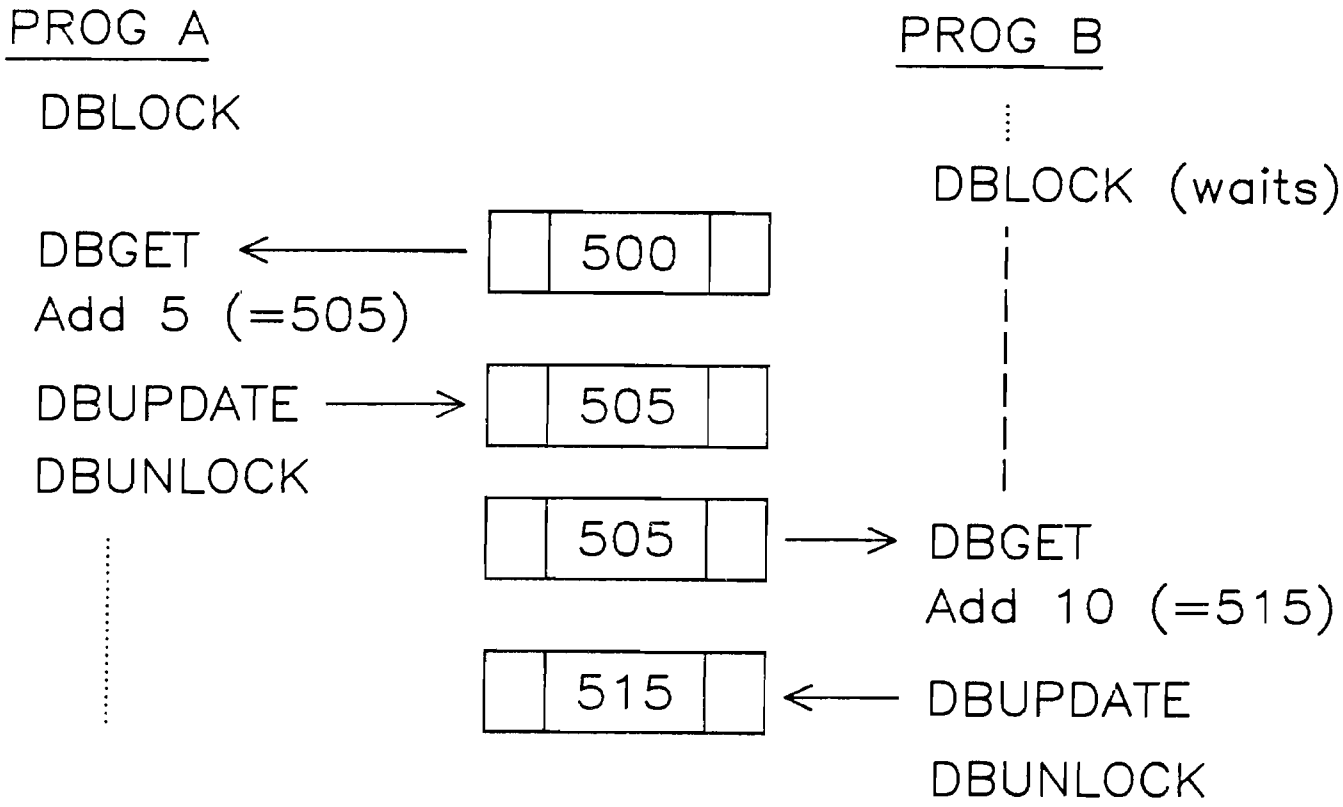


HEWLETT PACKARD

notes:

references:

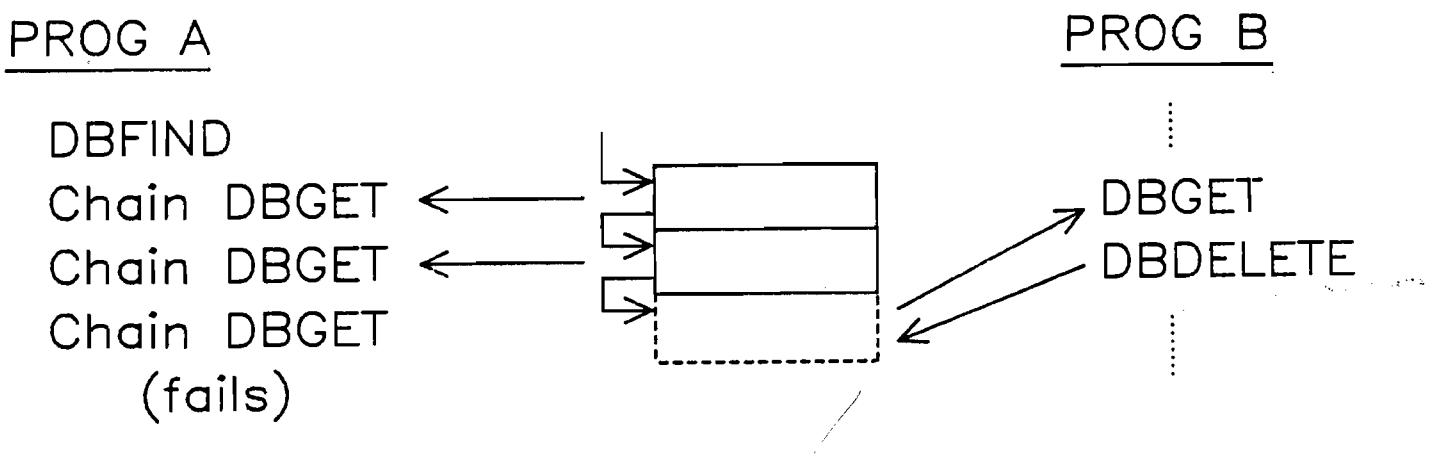
UPDATE WITH LOCKING



notes:

references:

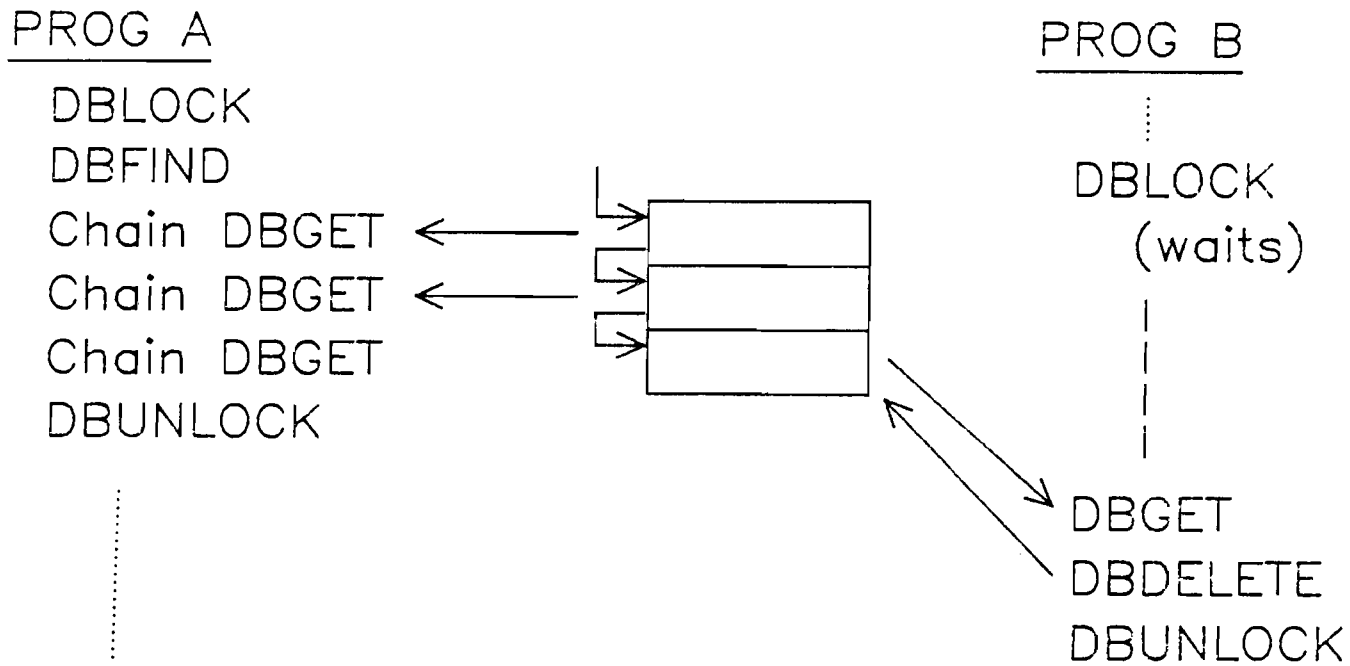
READING WITHOUT LOCKING



notes:

references:

READING WITH LOCKING



notes:

references:

A LOGICAL TRANSACTION IS:

* Basic unit of work performed against a data base

* Data is logically consistent before and after the transaction, but not while it is in progress

notes:

references:

LOCKING STRATEGY

* Locking is related to a logical transaction

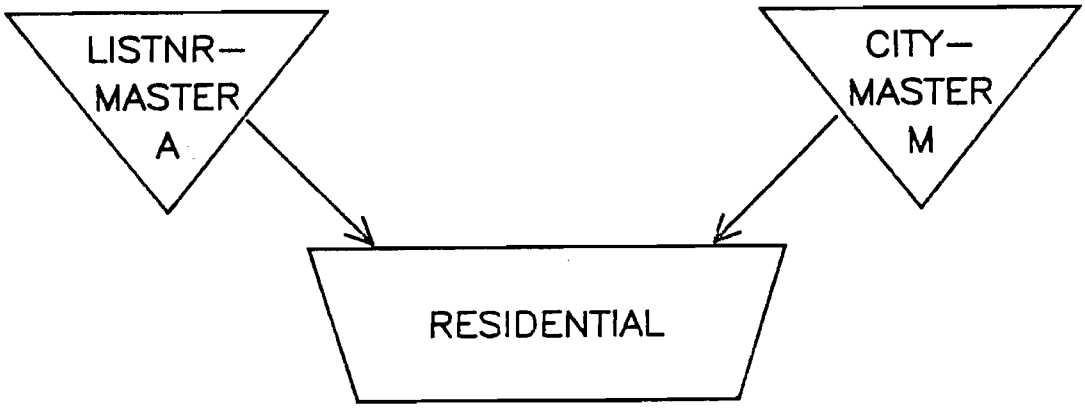
* At the start of any transaction, establish locks that cover:

  (1) all data entries you intend to change, and
  (2) all data entries which must not change during the transaction

notes:

references:

TRANSACTION



Example:

WONDER gets to list a new house

- Add LISTING–NR to CITY–MASTER
- Add RESIDENTIAL entry

notes:

references:

LOCKING EXAMPLE

* Transaction begins

```
CALL "DBLOCK" USING BASE, QUALIFIER ...
CALL "DBGET" USING BASE, CITY-MASTER ...
ADD 1 TO LISTING-NR
CALL "DBUPDATE" USING BASE, CITY-MASTER ...
CALL "DBPUT" USING BASE, RESIDENTIAL ...
CALL "DBUNLOCK" USING BASE ...
```

* Transaction ends

notes:

references:

DBLOCK

LOCKING LEVELS

* Data base
* Data set
* Data entry(s)

3 locking levels.

notes:

There are 6 modes in DBLOCK —
modes or different calls have totally
different meanings. Read mode is set in
DBOPEN

references:

DBLOCK

LOCKING CONTROL

* Unconditional locks

  return control after specified entity has been locked — may involve a wait

* Conditional locks

  return control immediately, indicating success or failure

notes:

references:

DBLOCK
(base, qualifier, mode, status)

* Data base
  - Mode 1 (unconditional)
  - Mode 2 (conditional)

* Data set
  - Mode 3 (unconditional)
  - Mode 4 (conditional)

* Data entry(s)
  - Mode 5 (unconditional)
  - Mode 6 (conditional)

notes:

references: IMAGE Reference Manual, Section V, DBLOCK
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DBLOCK

* Modes 1 and 2 (data base)
  - Qualifier is ignored
  - Ex:
```
CALL "DBLOCK" USING BASE, DUMMY,
     MODE1, STATS.
```

* Modes 3 and 4 (data set)
  - Qualifier contains set name or number
  - One data set allowed
  - Ex:
```
MOVE "CITY-MASTER;" TO SET.
CALL "DBLOCK" USING BASE, SET,
     MODE3, STATS.
```


notes:

```
          A      A or I      I       A
DBLOCK (base, qualifier, mode, status)
```

references:

DBLOCK

* Modes 5 and 6 (data entry)
  - Lock descriptor array in qualifier parameter
  - Qualifier may contain multiple descriptors

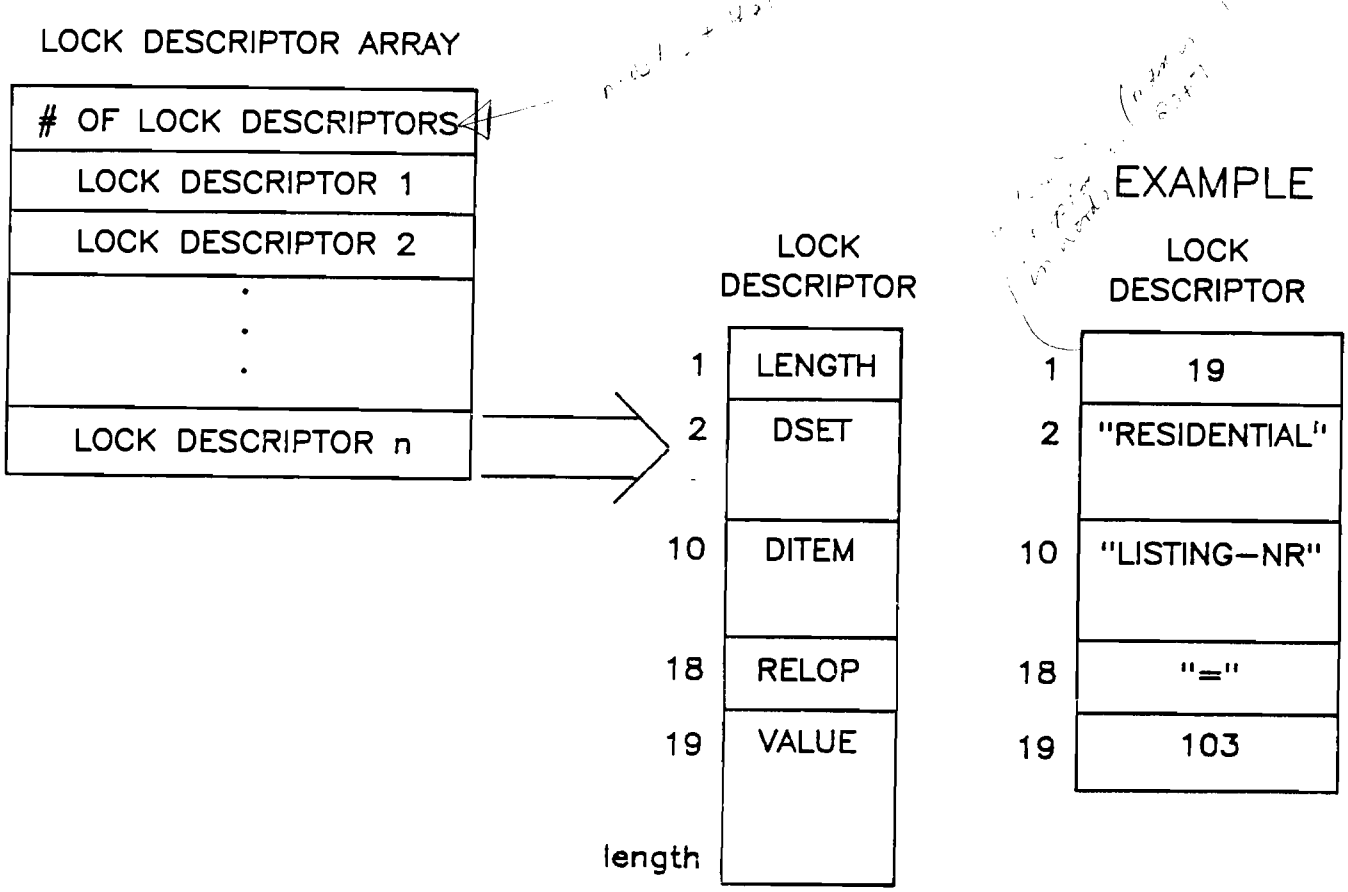
notes:

```
          A       A or I       I        A
DBLOCK (base, qualifier, mode, status)
```

- Each descriptor contains:
  data set, data item, relop, value
- Relop (relational operator) is one of: <= >= =
- Data item = "@" locks entire data set
- Data set = "@" locks data base

references:

LOCK DESCRIPTOR ARRAY



notes:

Lock descriptor example:

```
01 LOCK-DESC-ARRAY.
   05 NUM-ARRAYS  PIC S9(4) COMP VALUE +2.
   05 LOCK-DESC-1.
      10 DESC-LEN PIC S9(4) COMP VALUE +19.
      10 DSET     PIC X(16) VALUE "RESIDENTIAL".
      10 DITEM    PIC X(16) VALUE "LISTING-NR".
      10 RELOP    PIC X(2)  VALUE "=".
      10 LOCK-VAL PIC S9(4) COMP VALUE +103.
   05 LOCK-DESC-2.
      .
      .
      .
```

references:

DATA ENTRY LOCKING

* IMAGE allows only one data item per data set to be used for locking at a given time
* IMAGE does allow multiple values of the data item locked at one time

notes:

references:

THEREFORE:

* use the same item (when possible) for data entry locking to a given data set

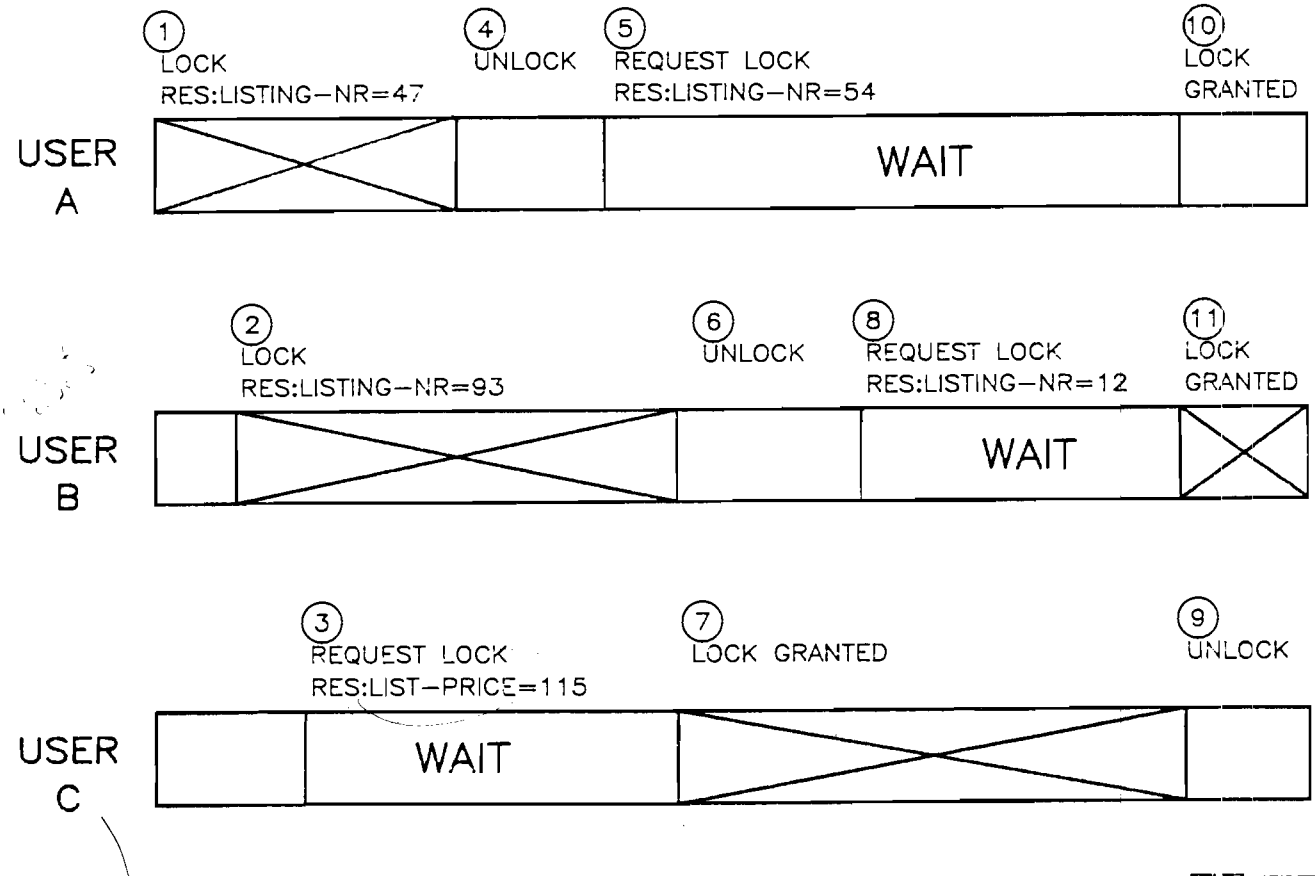
* for details, use item which is primary path's key

* for masters, use key item

notes:

references:

DATA ENTRY LOCKING EXAMPLE



HEWLETT PACKARD

notes:

references:

RPG

```
       7        15    19     26     30                        53      60     66
F      RES      IC    F      90R    4AM
F                                                             KIMAGE  WONDERLC
F                                                             KDSNAMERESIDENTIAL
F                                                             KITEM   CITY-ABBR
```

* KIMAGE spec defines locking mode
  - in column 66
  - at open time

* Special open modes

notes:

references: RPG Reference Manual, Section IV, Data Base Management Group Fields

RPG LOCKING MODES

* RPG automatic

  B data base for duration
  S data set for duration
  1 data base per record
  9 data set per record
  R record locking

* User-controlled

  L locking enabled —

notes:

references: RPG Reference Manual, Section IV, Locking Modes

RPG USER CONTROLLED LOCKING

```
6     10        18          28      33                54 56 58
C               CITKEY      LOCK    RES               737475
C               CITKEY      CHAINRES                  8058
C               CITKEY      UNLCKRES                     76
```

* Lock at data entry, data set, or data base level
* Conditional or unconditional locking
* Use LOCK and UNLCK operation

notes:

references: See APPENDIX D for further details on RPG & LOCKING

DBUNLOCK
(base, dset, mode, status)

* Releases all previously held locks of the calling process

* Mode – always a 1

* Dset – ignored, but still required

notes:

references: IMAGE Reference Manual, Section V, DBUNLOCK
MPE Pocket Guide, Section IV, IMAGE Intrinsics

LOCKING STRATEGY

USER DIALOGUE

* Don't lock around, instead ...

  GET
  dialogue
  LOCK
  RE-READ
  UPDATE
  UNLOCK

* If must lock around, use data entry locking

notes:

references:

LOCKING STRATEGY

LOCKING LEVELS

* If transaction is brief, lock at higher level
* Otherwise, to improve concurrency, lock at lowest possible levels (guidelines follow)

notes:

references:

LOCKING GUIDELINES

a. Chain DBGET – lock all data entries in the chain.

b. Serial DBGET – lock the data set.

c. Directed DBGET – lock the data set before determining relative record number.

d. Update – lock the data entry before calling DBGET.

e. Add to or delete from a detail data set – any lock which covers the data entry.

f. Add to or delete from a master data set – lock the data set or data base.

notes:

references:

CONCURRENT LOCKING

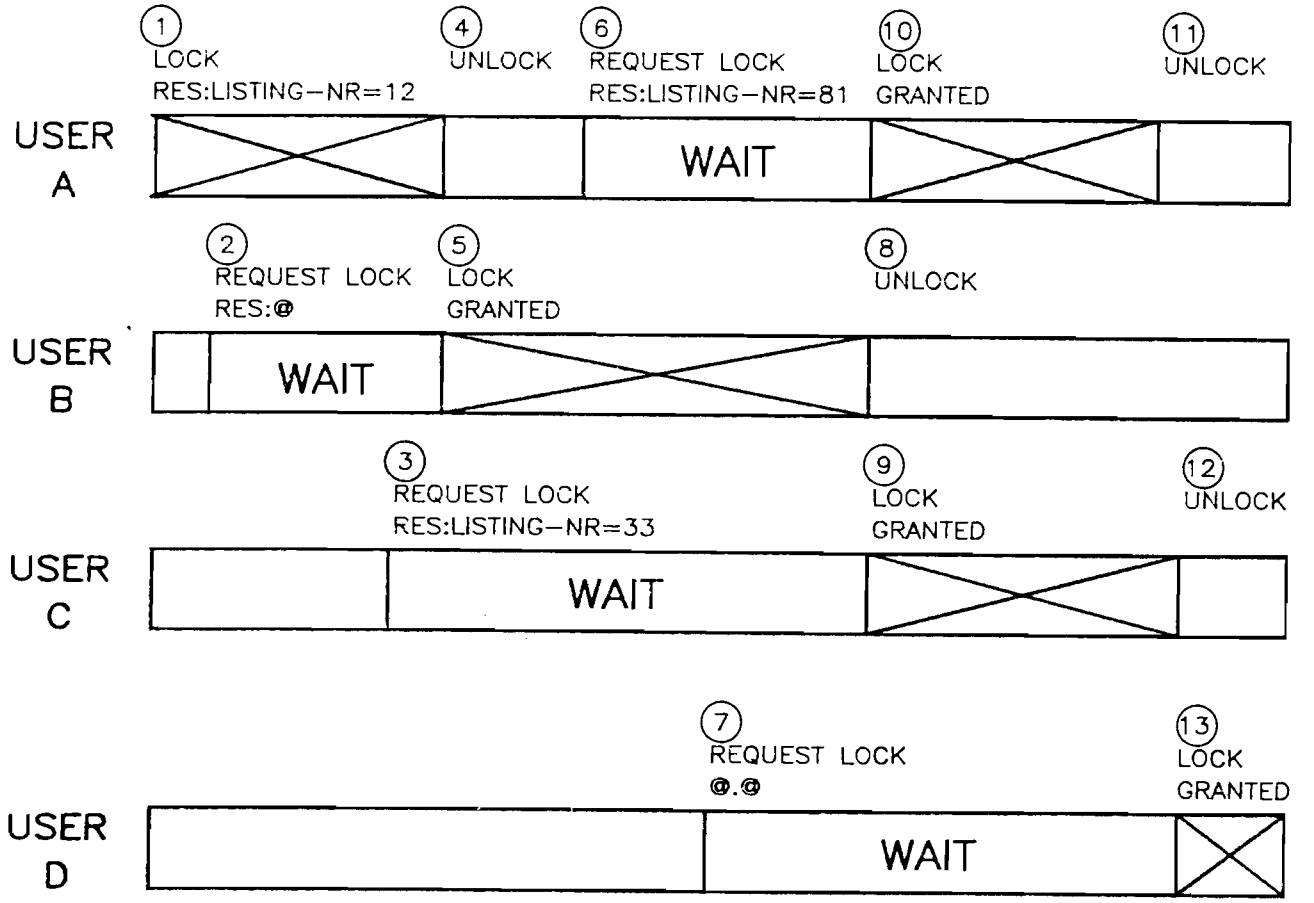
* Concurrent locking at different levels may cause conflict:
  - A data base lock cannot succeed until all locks of all levels are released
  - A data set lock cannot succeed until all data entries are released

* All lock requests are queued as a function of time

notes:

references:

LOCKING QUEUE EXAMPLE



notes:

references:

LOCKING CONSIDERATIONS

* DBOPEN mode 1
  - IMAGE requires a covering lock for all changes to data
  - Lock the set when modifying a master

* DBOPEN mode 3 or 7
  - Exclusive access – locking unnecessary

* DBOPEN mode 8
  - Read–only access – locking unnecessary

notes:

references:

LAB 8

DATA BASE ACCESS

WITH MULTIPLE USERS

notes:

references:

SECURITY

4.2 — 2.11 →

notes:

references:

DATA BASE SECURITY

* Purpose: to verify the user's right to use the data base

* Provided by hierarchy of security
  - MPE
  - DBOPEN procedure
  - IMAGE user class numbers

notes:

references:

HIERARCHY OF SECURITY

* MPE
  - Logon password protection
  - File access
  - Privileged file protection

* DBOPEN procedure
  - Password establishes user class number
  - Mode specifies modify, update or read-only access

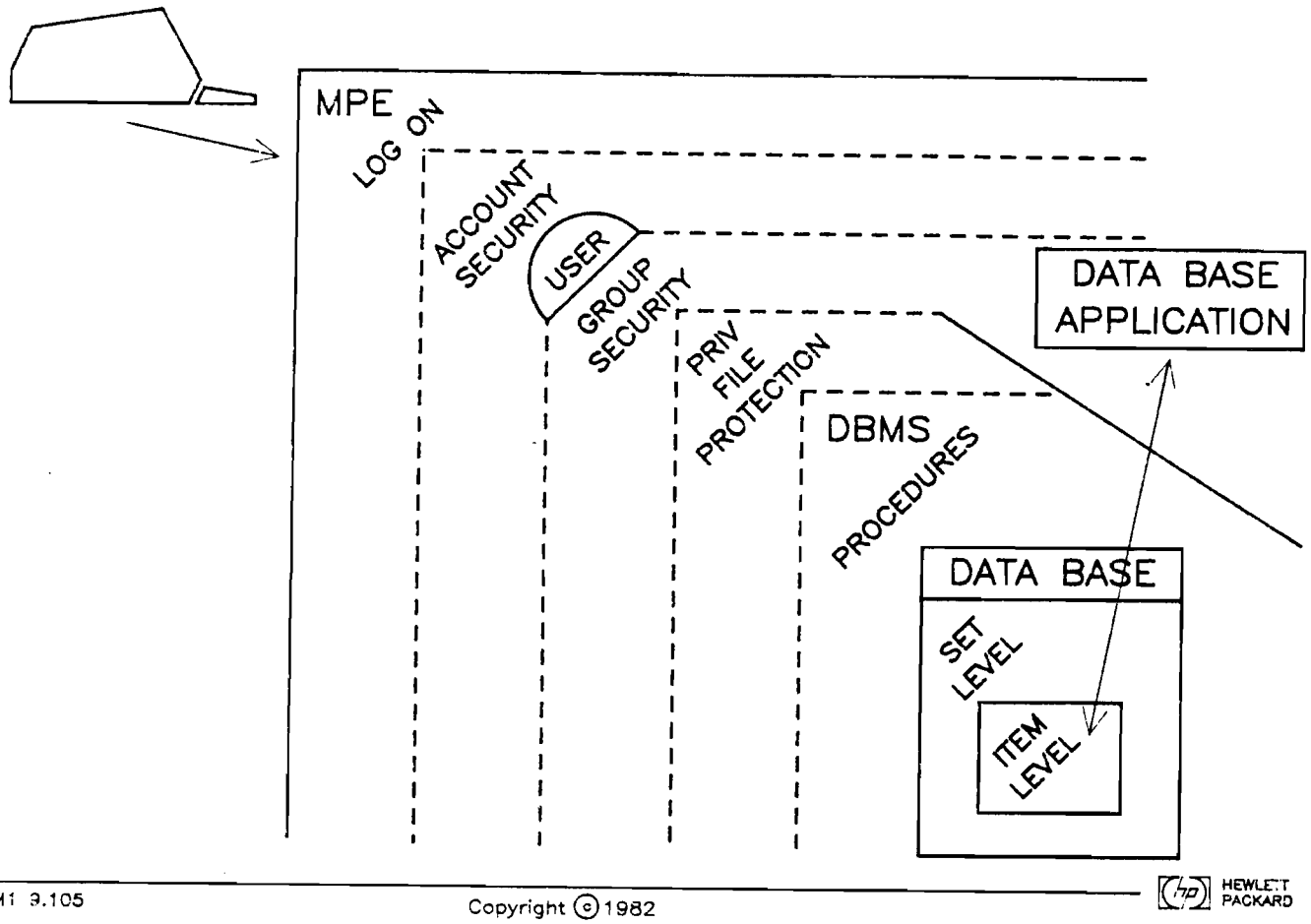
* IMAGE user class numbers
  - Specified at set and item levels
  - Indicates modify, update or read-only access

notes:

```
                A         A         I         IA
DBOPEN (base, password, mode, status)
```

references:

MPE/IMAGE OVERALL SECURITY



notes:

- Both MPE and IMAGE work together to provide privacy, security, and availability of data.

references: IMAGE Reference Manual, Section II, Protection of the Data Base

EXTERNAL SECURITY CONSIDERATIONS

* Store and sysdump tapes
  - Passwords and ASCII data readable

* MPE RESTORE
  - May accidentally restore data base files

* User capabilities
  - SM, AM, OP, PM

* Privileged file protection
  - No MPE or file system access

notes:

references:

IMAGE SECURITY
READ/WRITE CLASS LISTS

* If not on read/write class list, no access

* Write list implies read list

* Null class list – omitted
  - All users have read access
  - No users have write access

* Empty class list – (/)
  - Read and/or write list empty

notes:

references:

USER CLASS NUMBERS

* Creator (with password = ;) overrides all security
  - User class number is 64

* Wrong password assigns user class number 0
  - May be used in schema class lists

notes:

references:

SET LEVEL

* Read class list
  - Specifies users allowed to access items in set
  - Must also pass each item's security

* Write class list
  - Specifies users allowed modify access
  - Overrides item level security
  - Only if DBOPEN'd in modify mode (1, 3, 4)

notes:

SETS:

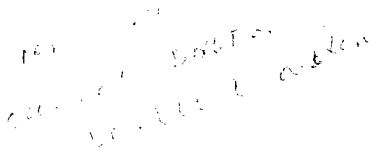
```
                {Manual   }
NAME: set name,{Automatic}[(read class list/write class list)];
                {Detail   }
```

references:

ITEM LEVEL

* Read class list
  - Specifies users allowed to read item

* Write class list
  - Specifies users allowed to update item
  - Only if DBOPEN'd in update mode (1, 2, 3, 4)
  - Can never update search or sort items

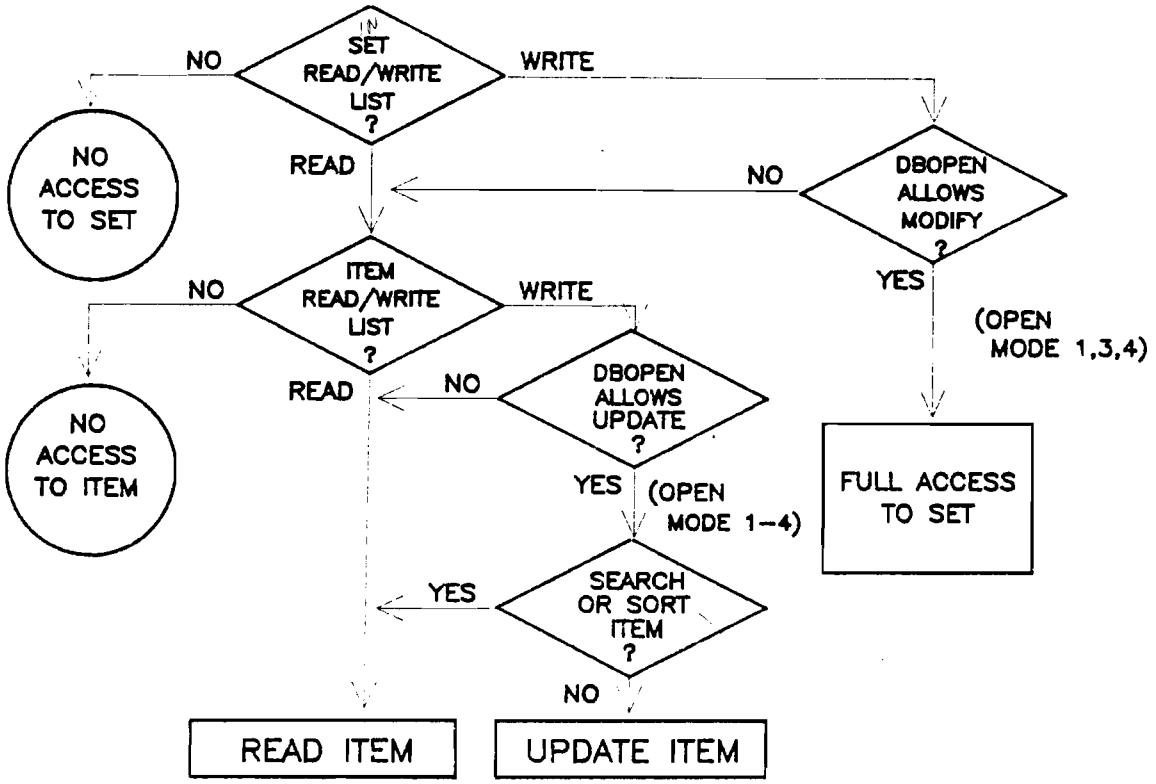


notes:

```
ITEMS:
  item name, ... [(read class list/write class list)];
```

references:

SECURITY FLOWCHART



notes:

references:

SECURITY WORKSESSION

* User runs program

  Password = SALESREP
  DBOPEN mode = 5 (Read)

* Resulting access — RESIDENTIAL

  Set level = Read
  Item level = Read all items

notes:

references:

SECURITY WORKSESSION

* User runs program

  Password = SALESREP
  DBOPEN mode = 1 ( )

* Resulting access — RESIDENTIAL

  Set level = _____
  Item level = _____
  _____

notes:

references:

SECURITY WORKSESSION

* User runs program

  Password = MANAGER
  DBOPEN mode = 5 (read only)

* Resulting access — RESIDENTIAL

  Set level = _____
  Item level = _____

notes:

references:

SECURITY WORKSESSION

* User runs program

  Password = MANAGER
  DBOPEN mode = 1 ( )

* Resulting access — RESIDENTIAL

  Set level = _____
  Item level = _____
  _____

notes:

references:

SECURITY EXERCISE

PASSWORDS:
2 TWO;
4 FOUR;
6 SIX;
8 EIGHT;
10 TEN;

Read (R) = DBFIND, DBGET
Update (U) = DBFIND, DBGET, DBUPDATE
Modify (M) = DBFIND, DBGET, DBUPDATE, DBPUT, DBDELETE

Note: "M" may only be specified at the Data Set level
Assume: Data Base opened in Mode 3 (Exclusive Access)

		0	2	4	6	8	10	CR
Master DSETA (2,4,6,8/2,4)			M	M	R	R		M
ITEM-1	(2,4/2,4)		U	U				U
ITEM-2	(/4,6)		U	U	U			U
KEY-3	(2,8/2,8)		R	R		R		R
ITEM-4	(2,8/10)		U	U		R		U
ITEM-5	(4,6/2,8)		U	U	R	U		U
Master DSETB (/8,10)						M	M	M
KEY-6	(2/2)					U	U	U
ITEM-7	(8,10/8,10)					U	U	U
ITEM-8	(8/8)					U	U	U
ITEM-9	(4/6)					U	U	U
ITEM-10						U	U	U

HEWLETT PACKARD

notes:

references:

SECURITY EXERCISE (cont'd)

	0	2	4	6	8	10	CR
Detail DSETC (4,6/)			R	R			M
ITEM-11 (/4,6)			U	U			U
KEY-3 (2,8/2,8)							R
ITEM-12 (4,6/)			R	R			U
ITEM-13			R	R			U
Detail DSETD (2,10/2)		M				R	R
ITEM-14		U				R	U
KEY-6 (2/2)		R					R
ITEM-15 (/)		U					U
ITEM-16 (0/0)		U					U
Detail DSETE (0/0)	U						R
ITEM-17	U						U
KEY-6 (2/2)	R						R
ITEM-18 (0/0)	U						U
Detail DSETF (/)							M
ITEM-19							U
Detail DSETG	R	R	R				M
ITEM-20	U	U	U			U	U

notes:

references:

LAB 9

DATA BASE RESTRUCTURING :

ADDING SECURITY FEATURES

notes:

references:

LAB 9 -- DATA BASE RESTRUCTURING ADDING SECURITY FEATURES

1. INTRODUCTION

WONDER Realty Company now has a fully operational (if simplified) data base. A customer can now call and be given a list of all properties in a given city with a specific number of bedrooms or vice versa. Since the receptionist, salesrep and manager all have terminals, this service can be provided by each of them.

Recently, the manager has run into problems. Some of his houses have been sold by the owners who were contacted directly by potential buyers. These buyers had gotten the owner's name, address, and phone number from the receptionist without ever setting foot in WONDER REALTY's offices. Some form of security is needed to solve the problem. In addition, the manager found out that too many clerical errors were occuring. He desired to be the only one with the ability to add and delete entries.

2. OBJECTIVES

(1) Learn data set and data item security.

(2) Practice the steps involved in data base resturcturing. Due to time constraints, this will be a shortened sequence. The full sequence will be addressed in LAB 10.

3. INSTRUCTIONS

A. FILES:

LAB4SCHM.groupn	Complete schema file in EDITOR format
DBSCHEMA.PUB.SYS	HP utility program to create root file
DBUTIL.PUB.SYS	HP utility program to create data sets
LOADDBJ.groupn	Data base load job stream
REALTY.groupn	REALTY data base
SOLxxxS.LABS	Final solution program

B. DETAILED INSTRUCTIONS:

1. Text LAB4SCHM into the EDITOR. Modify the schema to include the following security:

   Passwords: 10 RECEPT, 20 SALESREP, 30 MANAGER.

   The receptionist may not access the current owner, addressm, and phone number fields, but may read all other data. The salesrep may read all data, but may not update any fields. The manager has total access to all data, and the ability to add and delete records.

   HINT: The EDITOR may be used in the following manner to simplify editing:

   CHANGE "; " to " (nn/nn,nn);" in fromline/toline

   (fromline, toline are the first and last linenumbers of the itemlist. nn/nn,nn are password class numbers -- use the right format. "; " includes the proper number of spaces after the semicolon.)

   Finally, correct the remaining lines individually.

2. Run DBUTIL.PUB.SYS,PURGE to purge the current Data Base.

3. Run DBSCHEMA.PUB.SYS with appropriate file commands, parameters, to create the new REALTY root file.

4. Run DBUTIL.PUB.SYS,CREATE to create the remainder of the data base structure.

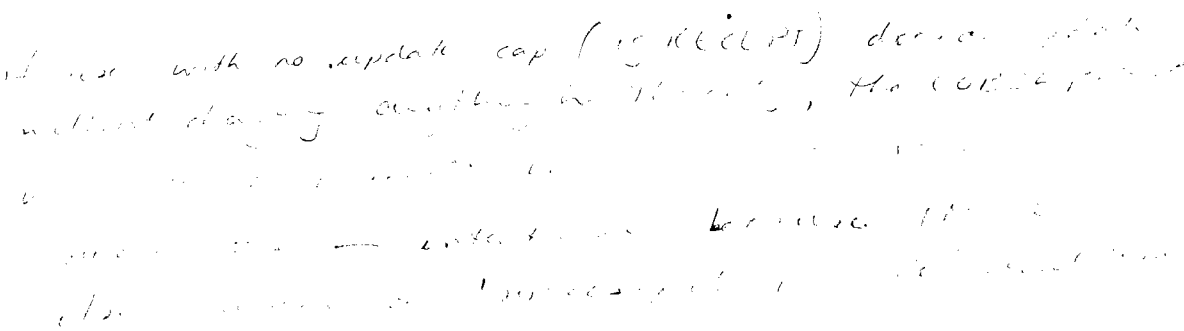
5. Stream LOADDBJ to load initial data into your data base.

6. Copy SOLxxxS.LABS to your logon group:
   FCOPY FROM=SOLxxxS.LABS;TO=SOLxxxS;NEW

7. Compile, prep, and run SOLxxxS.

8. You will be prompted for passwords for DBOPEN. Start with the manager password. Exit the program, then rerun, using the salesrep password. Exit the program, then rerun using the receptionist password. Observe the differences in displays: Did the receptionist get the owner's name, address, and phone? Were the salesrep and receptionist able to use the update functions?

C. SOLUTION:
The solution to this lab can be found in LAB9.SOLUTION and APPENDIX A-3
Please refer to this AFTER you have attempted the lab on your own.



UTILITIES AND RESTRUCTURING

TRANSACTION LOGGING AND RECOVERY

notes:

references:

IMAGE

UTILITIES

notes:

references:

UTILITIES

* DBUTIL – performs maintenance functions
* DBSTORE – stores data base
* DBRESTOR – copies data base to disc
* DBUNLOAD – copies data offline
* DBLOAD – loads data into data base
* DBRECOV – recovers data base

notes:

references: See Appendix E for IMAGE Utility Error Messages

DBUTIL

>>CREATE – build data base
>>PURGE – purge data base
>>ERASE – re-initialize data

>>HELP – help facility

>>ACTIVATE
>>DEACTIVATE
>>VERIFY
– used for accessing remote data bases over DSN/DS via data base access file

>>EXIT – exit DBUTIL

HEWLETT PACKARD

notes:

references: IMAGE Reference Manual, Section VIII, DBUTIL

DBUTIL

>>ENABLE
>>DISABLE
— data base name[/maint word]
FOR option[,option...]

* Options:

ACCESS — user access to data base

DUMPING — dumps user stack and DBCB if IMAGE procedure aborts

LOGGING — data base logging facility

RECOVERY — data base recovery facility

ILR — intrinsic level recovery

notes:

references: IMAGE Reference Manual, Section VIII, DBUTIL ENABLE/DISABLE

DBUTIL

>>SET data base name[/maint word]

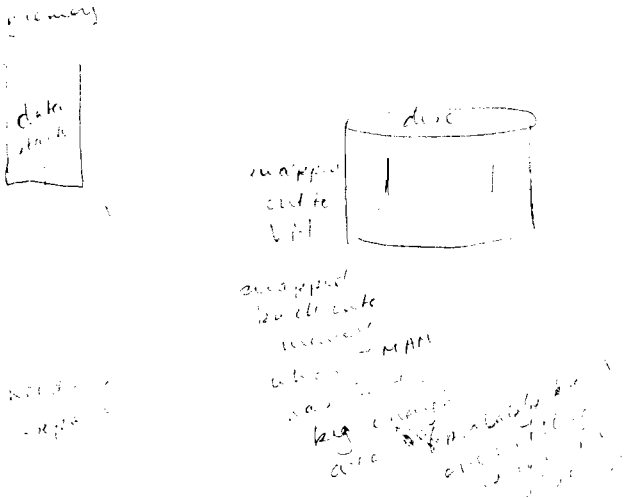
MAINT = maintenance word

BUFFSPECS = num buffers (from-users/to-users)[,num buffers (from-users/to-users)]...

LOGID = log identifier

PASSWORD = classnum=[password]

notes:



references: IMAGE Reference Manual, Section VIII, DBUTIL SET

DBUTIL

>>SHOW data base name [/maint word]

PASSWORDS
ALL
MAINT
BUFFSPECS
LOCKS
USERS
LOGID
FLAGS

[OFFLINE]

notes:

references: IMAGE Reference Manual, Section VIII, DBUTIL SHOW

* SHOW EXAMPLE

```
:RUN DBUTIL.PUB.SYS

>>SHOW REALTY ALL
   For data base REALTY

   MAINTENANCE WORD: ;;

   Access is Enabled
   Dumping is Disabled
   Logging is Disabled
   Recovery is Disabled

   LOGID:   REALOG   is valid
            password is correct

   BUFFER SPECIFICATIONS:
   8(1/2),9(3/4),10(5/6),11(7/8),12(9/10),13(11/12),14(13/14),
   15(15/16),16(17/18),17(19/120)

   No other users are accessing the data base

>>EXIT

END OF PROGRAM
```


notes:

references:

DBUTIL

>>RELEASE data base name

- suspends file system security for data base
- other groups and accounts have read, write and lock access
- IMAGE security left intact
- priv file security left intact

>>SECURE data base name

- restores security suspended by >>RELEASE

*** creator only ***

notes:

references: IMAGE Reference Manual, Section VIII, DBUTIL RELEASE/SECURE

DBSTORE

* Stores data base root file and data sets to tape or serial disc

* Sets timestamp and STORE flag in root file

* Format conforms to MPE STORE

* May be read by MPE RESTORE

```
:RUN DBSTORE.PUB.SYS
WHICH DATA BASE? REALTY
DATA BASE STORED
END OF PROGRAM
```


notes:

references: IMAGE Reference Manual, Section VIII, DBSTORE

DBRESTOR

* Restores data base root file and data sets from tape or serial disc

* Created by DBSTORE, MPE STORE or SYSDUMP

```
:RUN DBUTIL.PUB.SYS
>>PURGE REALTY
DATA BASE PURGED
END OF PROGRAM

:RUN DBRESTOR.PUB.SYS
WHICH DATA BASE? REALTY
DATA BASE RESTORED
END OF PROGRAM
```


notes:

references: IMAGE Reference Manual, Section VIII, DBRESTOR

DBUNLOAD

* Copies data entries from each data set to tape or serial disc
* No Root file or chain info copied
* Specially formatted
* Chained or Serial mode
  - Chained (Default)
    copies detail data entries by primary path
  - Serial
    copies all data entries by reading serially
* CTL(Y) monitors number of transactions

notes:

references: IMAGE Reference Manual, Section VIII, DBUNLOAD

DBUNLOAD

* If operating in chain mode and IMAGE detects a broken chain (primary path)
  - will copy entries up to break
  - then copy entries backward to break
  - will report broken chains and any missed entries

* :FILE DBUNLOAD=$NULL for chain check on primary path

notes:

references:

DBUNLOAD EXAMPLE

```
:RUN DBUNLOAD.PUB.SYS

WHICH DATA BASE? REALTY
  DATA SET 1:  44 ENTRIES

  DATA SET 2:  1 ENTRIES

  DATA SET 3:  1 ENTRIES

  DATA SET 4:  1 ENTRIES

  DATA SET 5:  0 ENTRIES

  DATA SET 6:  3 ENTRIES

  DATA SET 7:  44 ENTRIES

  DATA SET 8:  0 ENTRIES

END OF VOLUME 1, 0 WRITE ERRORS RECOVERED
DATA BASE UNLOADED
END OF PROGRAM
```


notes:

references:

DBLOAD

* Reads DBUNLOAD tapes and copies data entries to set of same data set number
* Not copied to automatics
* Data entries may be truncated or padded with binary zeroes
* Used for restructuring data base or for compacting along primary path

notes:

references: IMAGE Reference Manual, Section VIII, DBLOAD

DBLOAD EXAMPLE

```
:RUN DBUTIL.PUB.SYS

HP32215B.02.06 IMAGE/3000: DBUTIL (C) COPYRIGHT HEWLETT-PACKARD CO 1978
>>ERASE REALTY
   Data base REALTY has been ERASED
>>EXIT

END OF PROGRAM
:RUN DBLOAD.PUB.SYS

HP32215B.02.05 IMAGE/3000: DBLOAD (C) COPYRIGHT HEWLETT-PACKARD CO 1978

WHICH DATA BASE? REALTY
DATA SET 1: AUTOMATIC MASTER
DATA SET 2: AUTOMATIC MASTER
DATA SET 3: AUTOMATIC MASTER
DATA SET 4: AUTOMATIC MASTER
DATA SET 5:  0 ENTRIES
DATA SET 6:  3 ENTRIES
DATA SET 7: 44 ENTRIES
DATA SET 8:  0 ENTRIES
END OF VOLUME 1, 0 READ ERRORS RECOVERED
DATA BASE LOADED
END OF PROGRAM
```


notes:

references:

DATA BASE

RESTRUCTURING

notes:

references:

SEQUENCE OF OPERATIONS

1. Editor – modify schema text
2. DBSTORE – insurance
3. DBUNLOAD – copy entries to tape
4. DBUTIL,PURGE – purge old data base
5. DBSCHEMA – build new root file
6. DBUTIL,CREATE – build data base
7. DBLOAD – copy data into data base

notes:

references:

WHAT DOES DBLOAD DO?

* All entries from old set are put into new set which has same set number

* Not loaded if new set is automatic master

* Truncation or padding of entries may occur

notes:

references:

SUPPORTED DESIGN CHANGES

launched by DBLOAD !!

* Password part
  - Add, change or delete passwords and user class numbers
* Item part
  - Add, change name, change security
  - Delete, change definition if unreferenced by data set
* Set part
  - NAME: change name, security, master type
  - ENTRY: add to/delete from end of list
  - CAPACITY: increase or decrease

HEWLETT PACKARD

notes:

references:

UNSUPPORTED DESIGN CHANGES

* Unsupported option PARM=1234
  - Add/delete sets at end of schema
  - Add or delete paths
  - Reload into different group or account

* User unload/load program
  - Add/delete set other than at end of schema
  - Insert items into entry
  - Change item size

notes:

```
:RUN DBLOAD.PUB.SYS;PARM=1234
```

references:

LAB 10

UTILITIES AND

RESTRUCTURING

notes:

references:

A. Indicate with the proper letter from the below list, the utilities and the subsystems needed in order to accomplish the indicated activities against your data base. Assume your data base has not been recently backed up.

CASE 1: Increase the capacity of a data set and change the passwords

Solution: _____

CASE 2: Your response time has been getting noticeably slower over the past week while adding, deleting and retrieving data entries from a detail data set that has a sorted chain as a primary path.

Solution: _____

CASE 3: Add a new item to the end of a data entry.

Solution: _____

CASE 4: Change the structure of your data base, for example: add a new manual master and change the size of a data item.

Solution: _____

CASE 5: Your data base was accidently purged. However, your trusty system manager backed up the whole system this morning.

Solution: _____

COMMAND LIST

A. :EDITOR (modify original schema)
B. :RUN DBUTIL.PUB.SYS,CREATE
C. :RUN DBUTIL.PUB.SYS,ERASE
D. :RUN DBUTIL.PUB.SYS,PURGE
E. :RUN DBSTORE.PUB.SYS
F. :RUN DBRESTORE.PUB.SYS
G. :RUN DBUNLOAD.PUB.SYS,SERIAL
H. :RUN DBUNLOAD.PUB.SYS,CHAINED
I. :RUN DBLOAD.PUB.SYS
J. :RUN DBSCHEMA.PUB.SYS
K. :RUN user written unload program
L. :RUN user written load program

B. INSTRUCTOR DEMO

C. SOLUTION
The solution to this lab can be found in LAB10.SOLUTION.
Please refer to this AFTER you have attempted the lab on your own.

TRANSACTION

LOGGING

notes:

references: IMAGE Reference Manual, Section VII, TRANSACTION LOGGING AND RECOVERY SYSTEM

WHY LOG?

Just imagine....

Time: 3:00 P.M.
Day: Friday, end of the quarter

...Your clerks have entered 5000 transactions since the last data base backup...

and then...

SYSTEM FAILURE

notes:

references:

DATA

...one of the most important assets of a company

* Responsibility for data belongs to the data base administrator

* IMAGE logging and recovery is an HP supported, effective tool for protection against data loss

notes:

references:

IMAGE
LOGGING AND RECOVERY...

...GIVES YOU:

* Simple means of writing transactions to log file

* Facility to recover logged transactions in case of lost or corrupted data

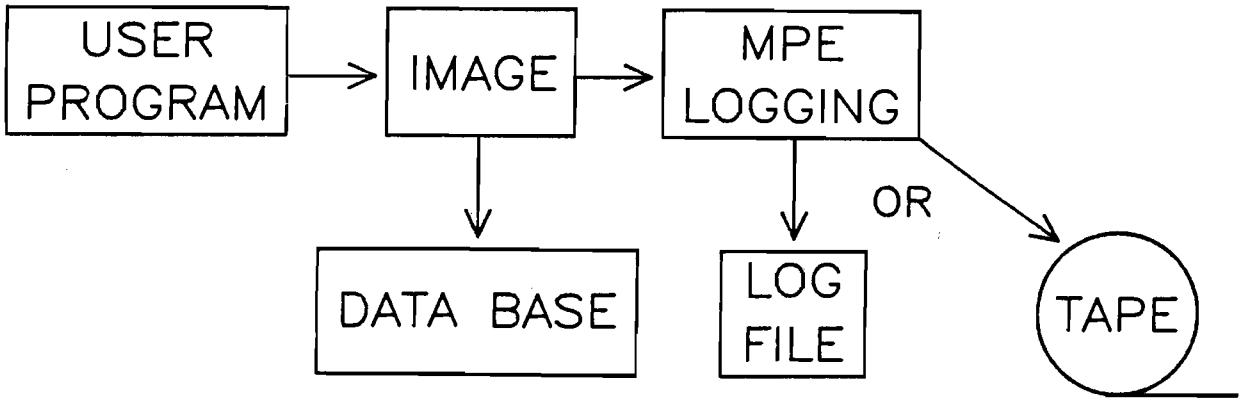
notes:

references:

LOGGING

* IMAGE passes all transactions that change the data base to MPE logging process

* MPE logging process writes transactions to log file associated with the data base



notes:

references:

WHAT IS LOGGED?

* Calls to:

  DBOPEN
  DBCLOSE
  DBPUT
  DBDELETE
  DBUPDATE
  DBBEGIN
  DBEND
  DBMEMO

* Calls to log intrinsics built into IMAGE procedures

notes:

references:

DBBEGIN – DBEND

* DBBEGIN and DBEND defines the start and end of a logical transaction

* If not used, each DBPUT, DBDELETE and DBUPDATE is a single logical transaction

notes:

```
DBBEGIN   A      A      I       A          I
DBEND    (base, text, mode, status, textlen)
DBMEMO
```

references: IMAGE Reference Manual, Section V, DBBEGIN, DBEND
MPE Pocket Guide, Section IV, IMAGE Intrinsics

DBBEGIN – DBEND

* Sequence of calls
  - call DBLOCK covering transaction
  - calls to locate data
  - call DBBEGIN
  - calls to modify data
  - call DBEND
  - call DBUNLOCK

notes:

references:

THE LOGGING CYCLE

The logging cycle runs from backup to backup. To determine logging cycle length, consider:

* high transaction volume $\Longrightarrow$ shorter cycle
* large data base $\Longrightarrow$ longer cycle
* data irretrievable $\Longrightarrow$ shorter cycle

notes:

references:

INITIALIZING LOGGING

1. Create the logid

   :GETLOG RLTYLOG;LOG=DBRLTYL,DISC

2. Place logid into root file

   :RUN DBUTIL.PUB.SYS
   >>SET REALTY LOGID=RLTYLOG

HEWLETT PACKARD

notes:

```
:GETLOG logid;LOG=logfile{,DISC}
                         {,TAPE}

>>SET data base name[/maint word] LOGID=log identifier
```

references:

INITIALIZING LOGGING

3. Set flags in root file

```
:RUN DBUTIL.PUB.SYS
>>ENABLE REALTY FOR LOGGING,RECOVERY
    WARNING: Data base modified and
    not DBSTORED
>>DISABLE REALTY FOR ACCESS
```

4. Back up the data base

```
:RUN DBSTORE.PUB.SYS
WHICH DATA BASE? REALTY
```


notes:

```
>>ENABLE }                                          {LOGGING }
        } data base name[/maint word] FOR          {RECOVERY}
>>DISABLE}                                         {ACCESS  }
```

references:

LOGGING CYCLE STARTUP

1. Prepare the log file
   - if logging to tape, mount the tape
   - if logging to disc, rename or purge old logfile, and build new one (same name)

```
:BUILD DBRLTYL;CODE=LOG;DISC=1600,8,5;DEV=4
```

do, at want to run out of disc space while be
- not sufficient

notes:

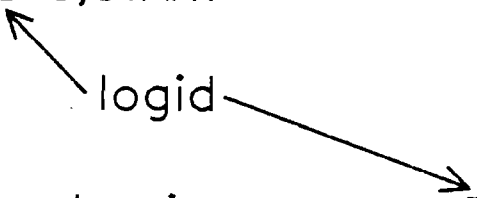
```
:BUILD logfile name;CODE=LOG;DISC=[numrec]
       [,[numextents][,initalloc]];DEV=device
```

references:

LOGGING CYCLE STARTUP

2. Start MPE log process from master console

```
:LOG RLTYLOG,START
```




```
8:00/6/user logging process RLTYLOG is running
```

3. Set flags in root file

```
:RUN DBUTIL.PUB.SYS
>>ENABLE REALTY FOR ACCESS

>>DISABLE REALTY FOR RECOVERY
```


notes:

```
                {START   }
:LOG logid,     {RESTART}
                {STOP    }
```

references:

LOGGING CYCLE SHUTDOWN

1. Stop all use of data base

   :RUN DBUTIL.PUB.SYS
   >>SHOW REALTY USERS
   NO OTHER USERS ARE ACCESSING THE DATA BASE

2. Set flags in root file

   :RUN DBUTIL.PUB.SYS
   >>DISABLE REALTY FOR ACCESS
   >>ENABLE REALTY FOR RECOVERY

3. Stop MPE log process from master console

   :LOG RLTYLOG,STOP

notes:

```
>>SHOW data base name[/maint word] USERS
```

references:

LOGGING CYCLE SHUTDOWN

4. Back up the data base

   :RUN DBSTORE.PUB.SYS
   WHICH DATA BASE? REALTY

5. Archive log files

   - If on disc, MPE STORE to tape
   - Place tapes (tape or disc log file) in vault

notes:

references:

LOGGING MEDIA

* Disc
  - Subject to system directory problems
  - If end of file is reached, recovery will be required
  - Allows concurrent access of logfile by DBRECOV

* Tape
  - Requires dedicated tape drive
  - If end of tape is reached, system will buffer and request next reel
  - Subject to power failure complications
  - Subject to tape parity errors

notes:

perhaps

references:

LOGGING TO DISC

* Place logfile on disc device other than that of data base (if possible)
* Reserve enough space to hold all log records in a logging cycle
* Allocate all required extents
* Use LISTF to monitor number of records in logfile
* Use LISTDIR2 to document logfile disc address

notes:

references: See Appendix F for calculation of number of log records

LOGGING TO TAPE

In the event of a power failure, stop all data base access and perform logging shutdown and startup procedures

notes:

references:

LOGGING CONSIDERATIONS

* Should be an integral part of computer systems operations

* Must have management backing

* Results in reliable data base integrity

* Requires time — should be planned for

* Additional HP assistance available

* Never process after a system failure without recovering first

notes:

references:

DATA BASE

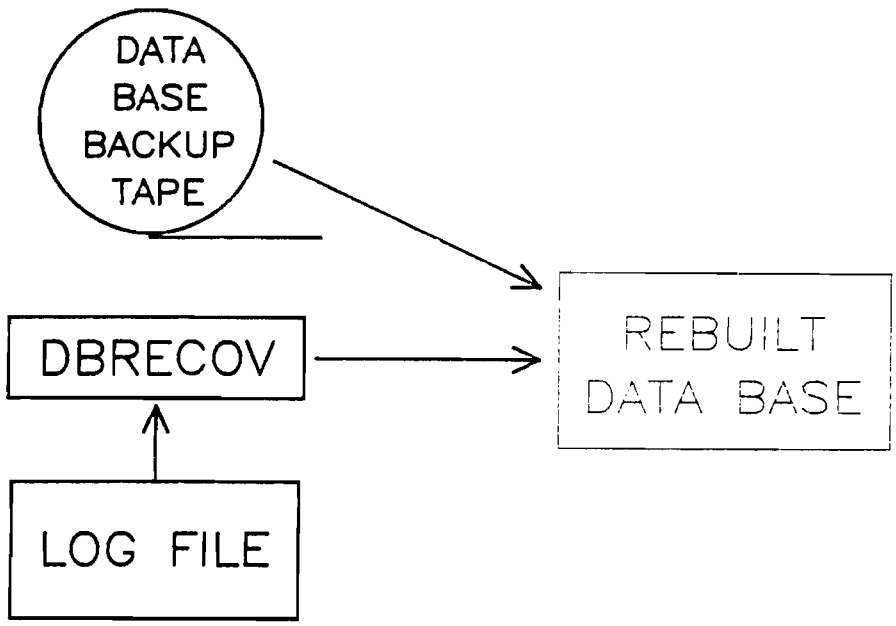
RECOVERY

notes:

references: IMAGE Reference Manual, Section VII, TRANSACTION LOGGING AND RECOVERY SYSTEM

DBRECOV

If a failure occurs, the last data base backup tape plus the log files provide data for DBRECOV to rebuild the damaged data base:



notes:

references:

DBRECOV COMMANDS

>RECOVER – defines data base(s)

>CONTROL – specifies conditions

>FILE – requests user recovery files

>PRINT – pre–recovery information

>RUN – performs actual recovery

notes:

references: IMAGE Reference Manual, Section VIII, DBRECOV
MPE Pocket Guide, Section IV, DBRECOV

RECOVERY PROCEDURES

1. Warmstart
   - recovers disc buffer if logging to tape
   - updates EOF if logging to disc
2. DBSTORE the data base (optional)
3. MPE STORE log file if on disc (optional)
4. DBUTIL,PURGE the data base
5. DBRESTOR the data base
   - use backup copy from beginning of current logging cycle

notes:

references:

RECOVERY PROCEDURES

6. DBUTIL SHOW data base FLAGS to verify
   - access is disabled
   - logging is enabled
   - recovery is enabled

7. RUN DBRECOV.PUB.SYS
   >RECOVER REALTY
   >FILE UFILE1,USER.IMAGE,1,0
   >RUN
   >EXIT

notes:

>>SHOW data base name[/maint word] FLAGS

>FILE fileref,useref[,rmode,fmode]

references:

RECOVERY PROCEDURES

8. DBSTORE the data base

9. Run user program against user log file(s) to determine last recovered transaction

10. Perform logging startup procedures and begin processing

notes:

references:

ALTERNATE RECOVERY PROCEDURE

* Recover data base with DBRECOV

* Due to time constraint, DBSTORE is not done
  - Log to new log file
  - Set flags in root file for access
  - Resume user access
  - In the event another recovery is performed:
    1. Recover from first log file
    2. Recover from subsequent log files with >CONTROL NOSTORE

notes:

references:

ILR
INTRINSIC LEVEL RECOVERY

* Insures data base structural integrity if DBPUT or DBDELETE could not complete its operation

* "Logging" invisible to user

* Recovery performed automatically

notes:

references:

ENABLING ILR

```
:RUN DBUTIL.PUB.SYS
>>Enable REALTY for ILR
    ILR is Enabled

:LISTF REALTY@
REALTY
REALTY00 <------- ILR file
REALTY01
REALTY02
REALTY03
```


notes:

references:

ILR CONSIDERATIONS

* Be aware that ILR only guarantees structural integrity – not logical

* IMAGE logging still strongly recommended

notes:

references:

LAB 11

TRANSACTION LOGGING

AND RECOVERY

notes:

references:

A. TRANSACTION LOGGING

1. Transaction logging can log to tape T F
   system domain disc T F
   serial disc T F
   private volume disc T F

2. IMAGE logging uses MPE user logging. T F

3. All IMAGE calls may produce a log record. T F

4. What IMAGE procedures define a logical transaction?

   _____ and _____

5. Without these procedures, no logging takes place. T F

6. It is best to include IMAGE locks within the logical transaction. T F

7. Briefly list 4 steps to initialize IMAGE logging the first time.

   _____, _____,

   _____, _____

8. When building a logfile on disc, which name is used?

   _____

9. When starting log process with LOG command, which logid is used?

   _____

10. What should be the status of the flags in the root file in order to process data base applications?

    _____, _____

11. How does one know when all users are off the data base?

    _____

12. What should be the status of the flags in the root file in order to back up the data base?

_____, _____

13. Why? _____

14. What method of backup is recommended? _____

15. Why? _____

16. How would one tell that a disc log file is approaching full?

_____

17. Why allocate all extents of a disc log file?

_____

B. TRANSACTION RECOVERY

1. What is the first step of recovery? _____

2. Purge the data base using DBUTIL before DBRESTOR? T F

3. Status of flags for recovery?

_____, _____,

_____

4. How does DBRECOV check for correct log files?

_____ in logfile must be _____

that of root file.

5. What command from DBRECOV executes recovery? _____

6. ILR maintains data base logical integrity T F
structural integrity T F

IMAGE

INTERNALS

notes:

references:

TOPICS

* IMAGE AS A SET OF FILES
* IMAGE RECORDS AND RECORD PLACEMENT
* IMAGE CONTROL BLOCKS
* INTERNALS OF DATA BASE PROCEDURES

notes:

references:

IMAGE

AS A

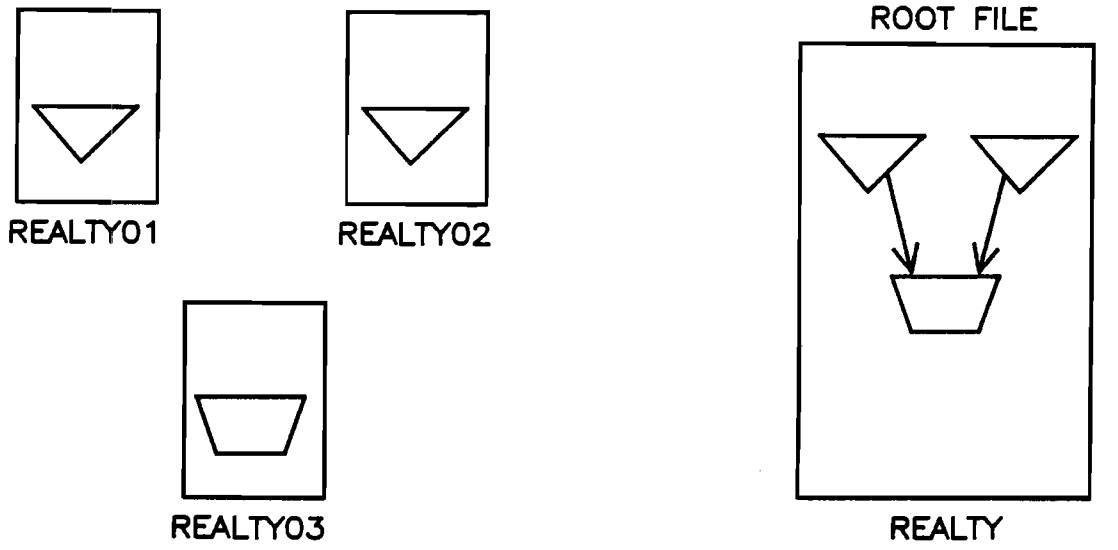
SET OF FILES

notes:

references:

* Each data set is a file

* Data sets are linked through the ROOT FILE which contains an "IMAGE" of the data base



notes:

references:

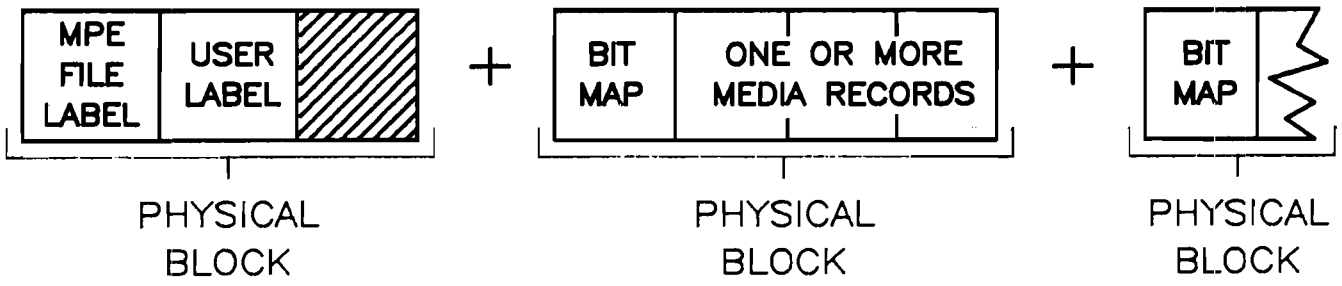
ROOT FILE CONTENTS

* Creation date and time
* Maintenance word
* Passwords
* Item table – name, type, length, etc.
* Set table – name, type
* Security for items and sets
* Data set control blocks – capacity, length, blocksize, counts, record definition, path tables

notes:

references:

MASTER DATA SET



USER LABEL = free space counter
data set capacity

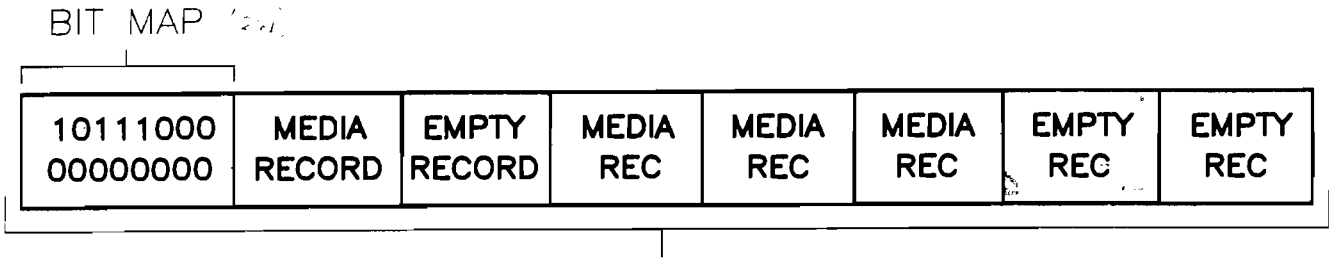
BIT MAP – one bit per record in block to indicate whether record is active or empty

HEWLETT PACKARD

notes:

references:

BIT MAP



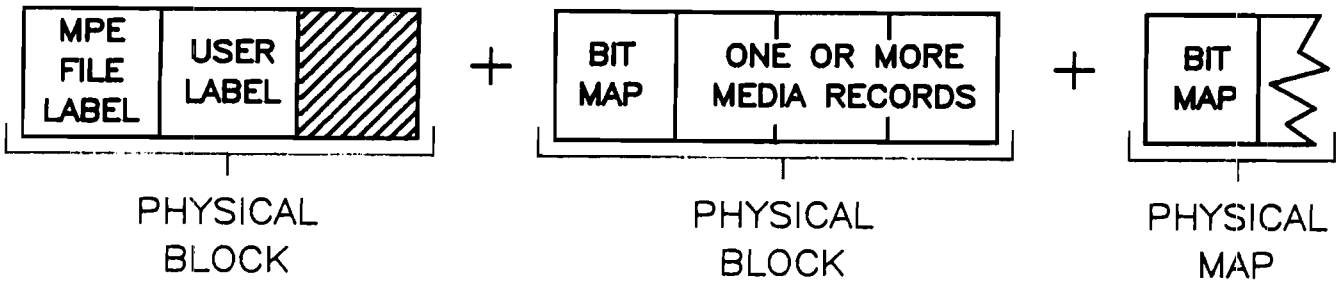
Block contains 7 records and 2nd, 6th, and 7th are empty

HEWLETT PACKARD

notes:

references:

DETAIL DATA SET



USER LABEL = free space counter (2w)
end of file pointer (2w)
delete chain pointer
- address of record to be deleted
- checked when adding a record

BIT MAP – same as master data set

notes:

references:

IMAGE RECORDS

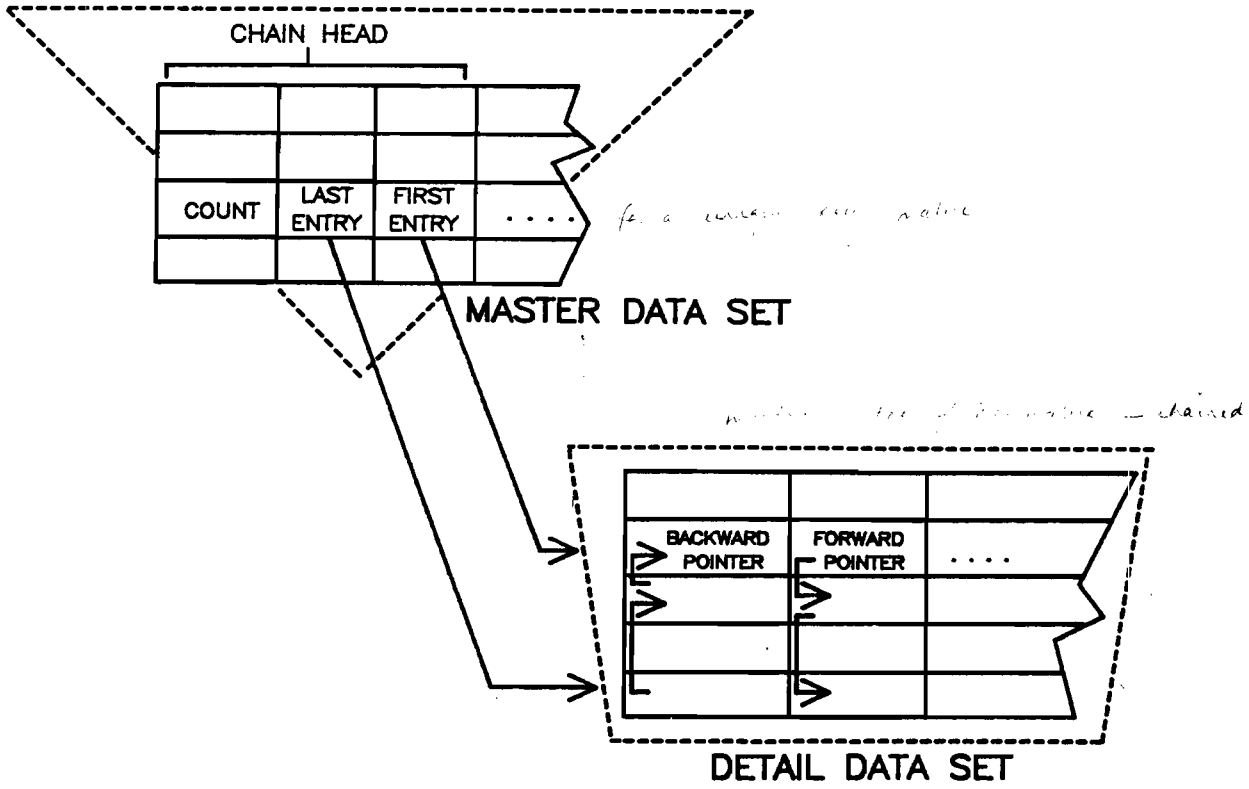
AND

RECORD PLACEMENT

notes:

references:

DATA SET RELATIONSHIPS

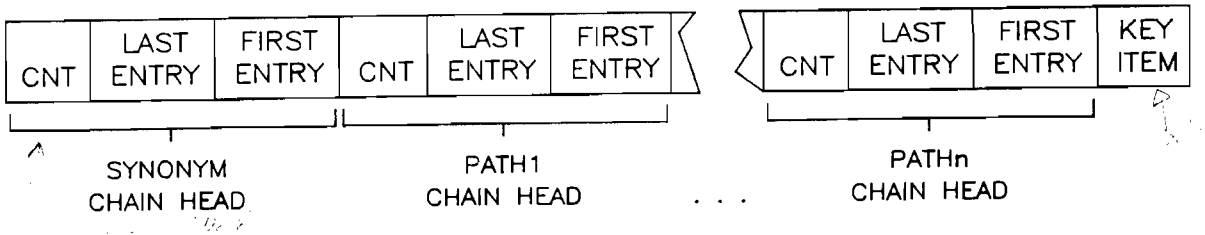


notes:

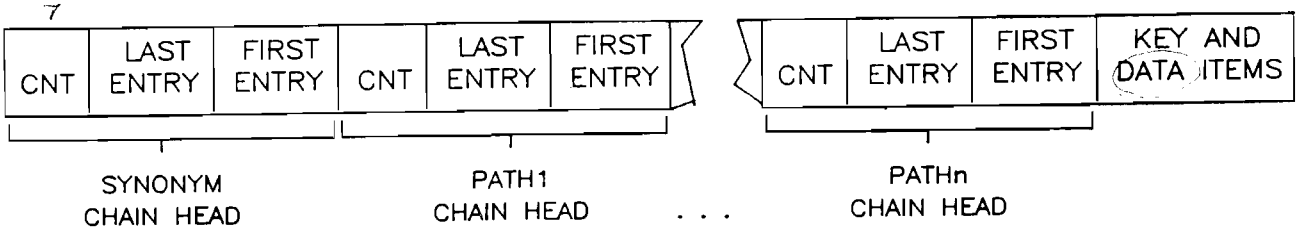
references:

MEDIA RECORD: POINTERS + DATA

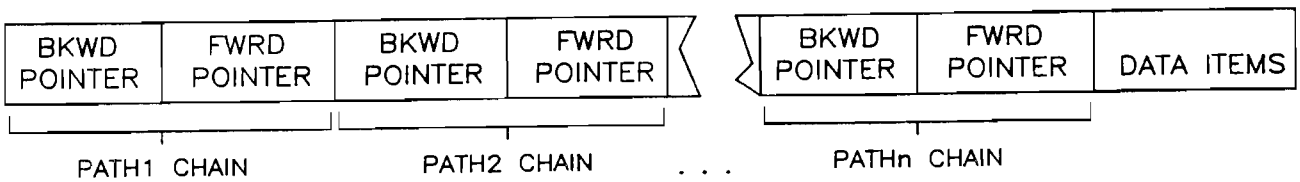
AUTOMATIC MASTER



MANUAL MASTER



DETAIL



HEWLETT PACKARD

notes:

- pointers require 2 words
- counts require 1 word

- chain head = 5 words
- chain = 4 words

references:

HASHING

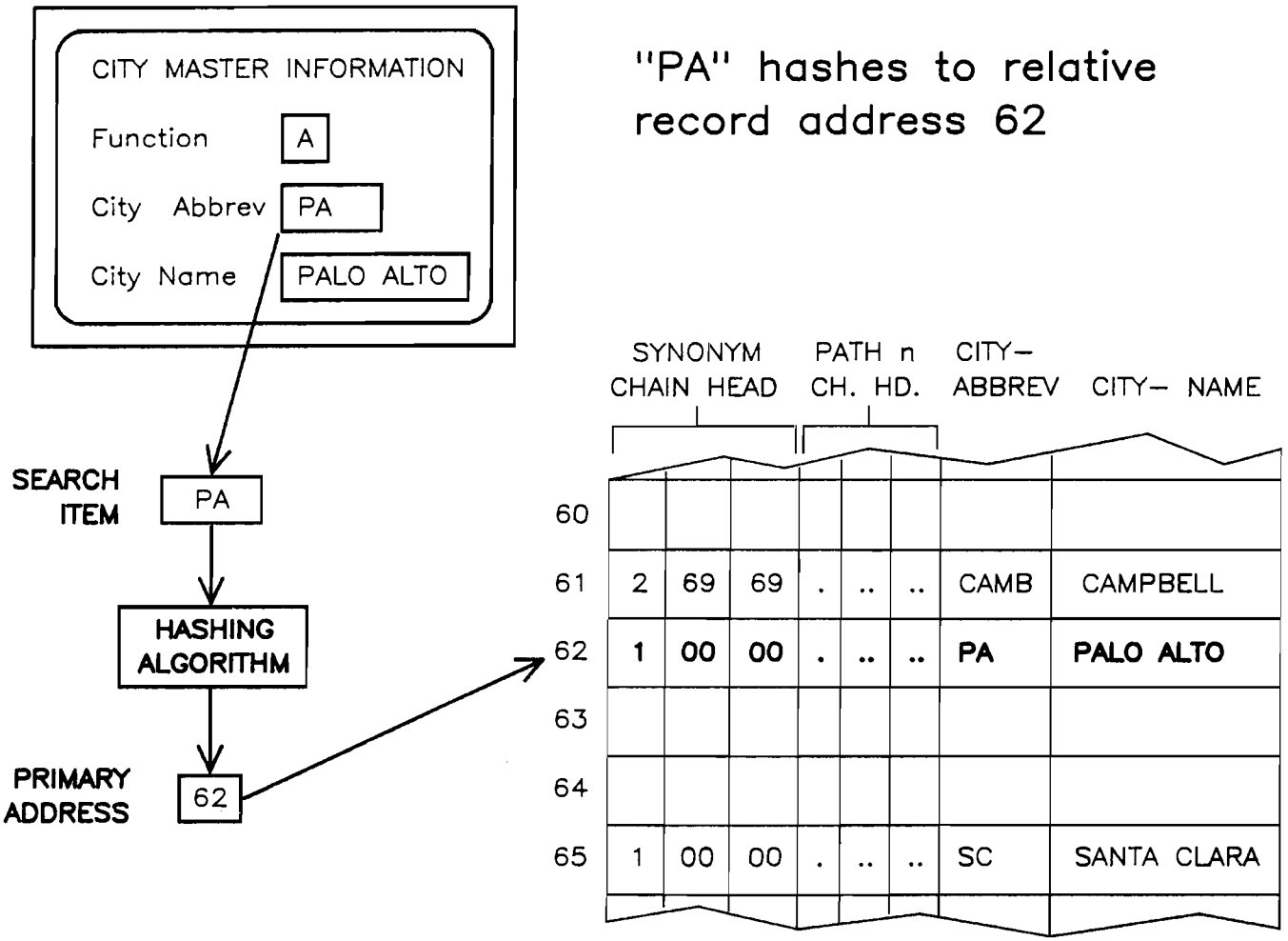
* Arithmetic algorithm applied to item value to produce a relative record address

* Used to place master's only!

* Master that gets stored at this address is known as a primary –

notes:

references: IMAGE Reference Manual, Section X, Primary Address Calculation



"PA" hashes to relative record address 62

notes:

references:

SYNONYM CHAINS

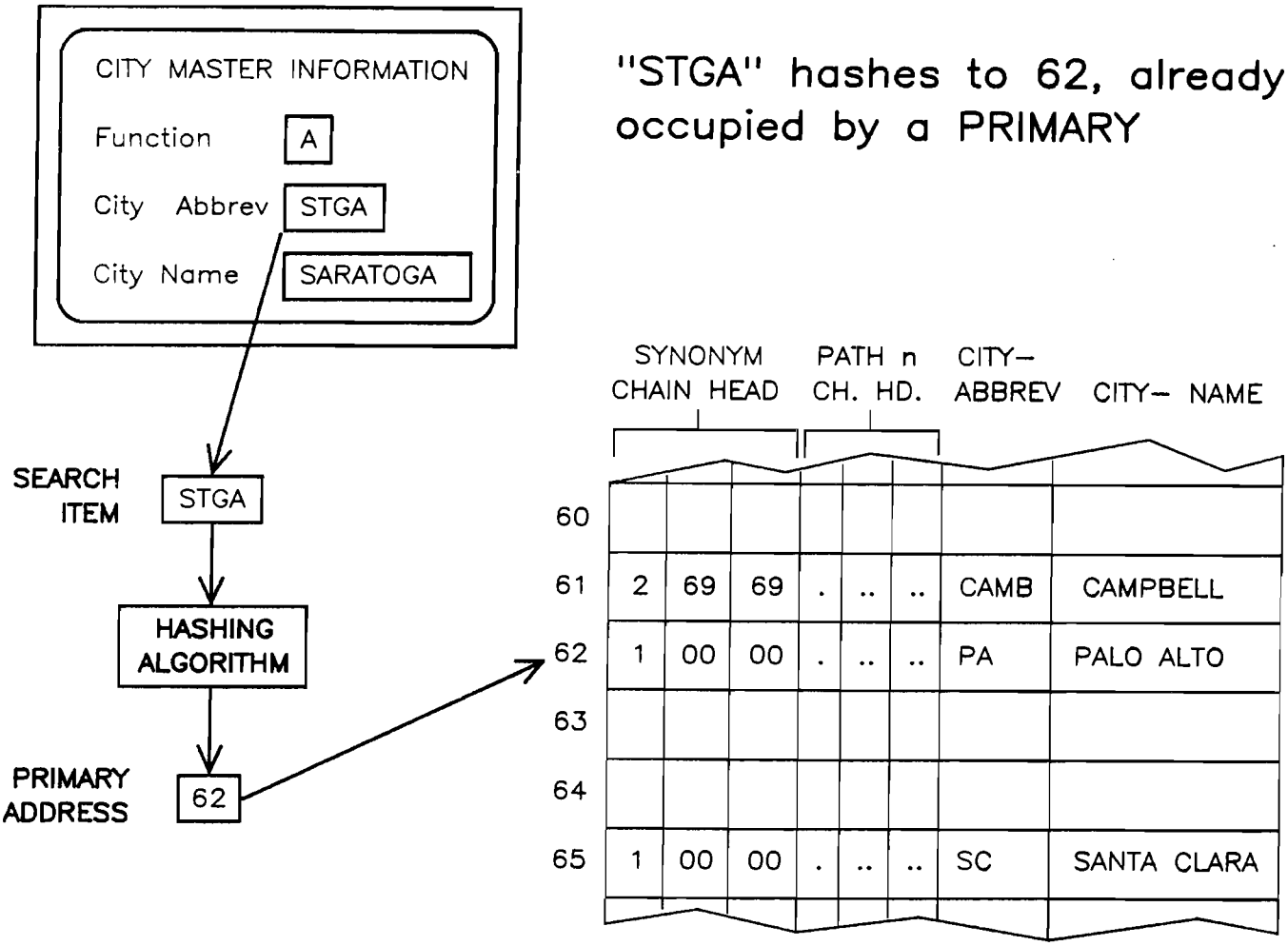
* Synonyms occur when different search item values hash to the same address

* IMAGE assigns a secondary address to the new record and it is now known as a SECONDARY

* Synonym chains consist of the first record to hash to an address (PRIMARY), and all the synonyms (SECONDARIES)

notes:

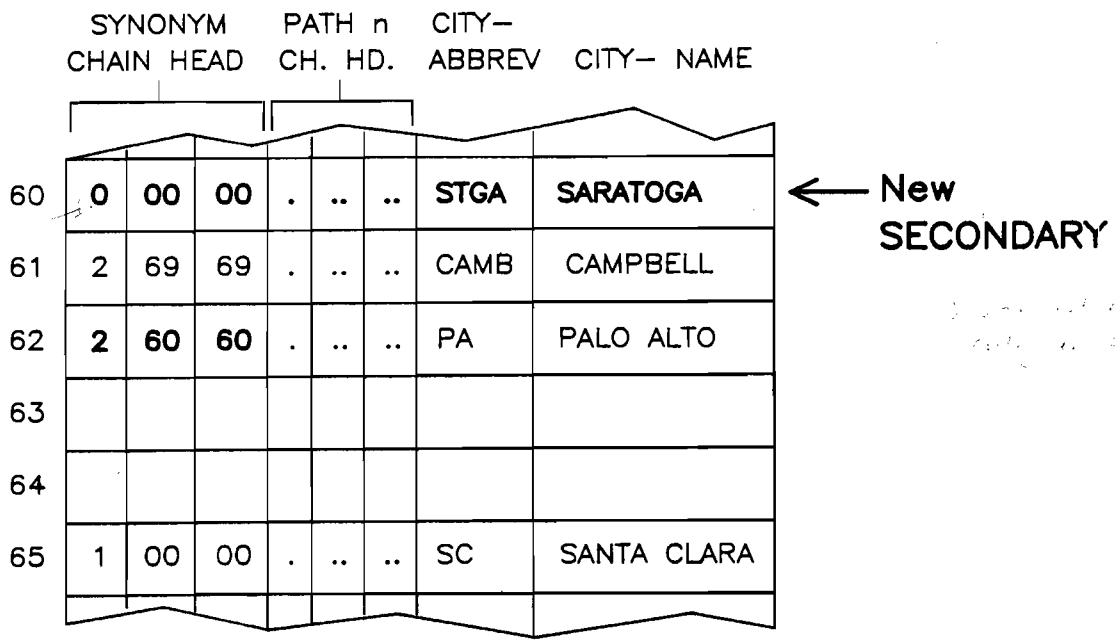
references:



notes:

references:

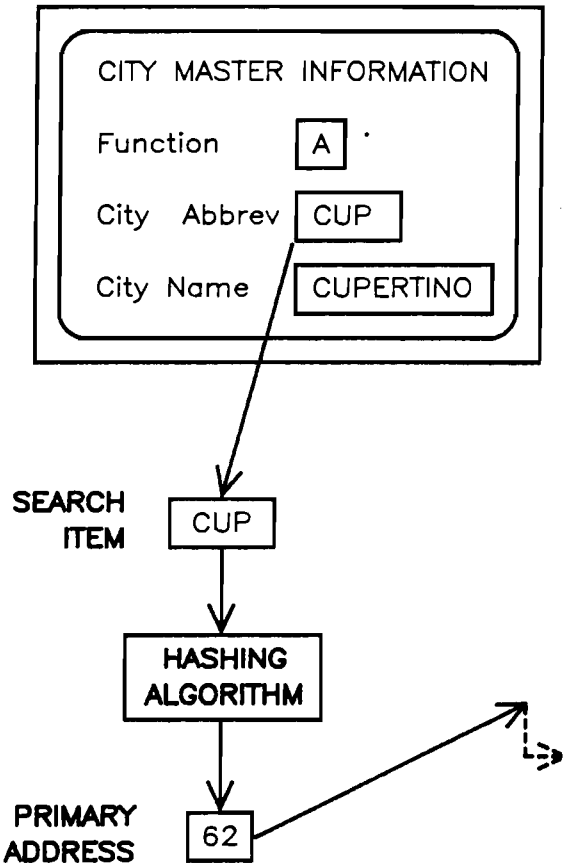
"STGA" is assigned a secondary address of 60



HEWLETT PACKARD

notes:

references:



"CUP" hashes to 62 and is assigned a secondary address of 63

	SYNONYM CHAIN HEAD			PATH n CH. HD.			CITY– ABBREV.	CITY– NAME
60	0	00	**63**	.	..	..	STGA	SARATOGA
61	2	69	69	.	..	..	CAMB	CAMPBELL
62	**3**	**63**	60	.	..	..	PA	PALO ALTO
63	**0**	**60**	**00**	.	..	..	**CUP**	**CUPERTINO**
64								
65	1	00	00	.	..	..	SC	SANTA CLARA

HEWLETT PACKARD

notes:

- count = 0 is SECONDARY
- count = 1 is PRIMARY with no SECONDARIES
- count > 1 is PRIMARY with SECONDARIES

references:

MIGRATING SECONDARIES

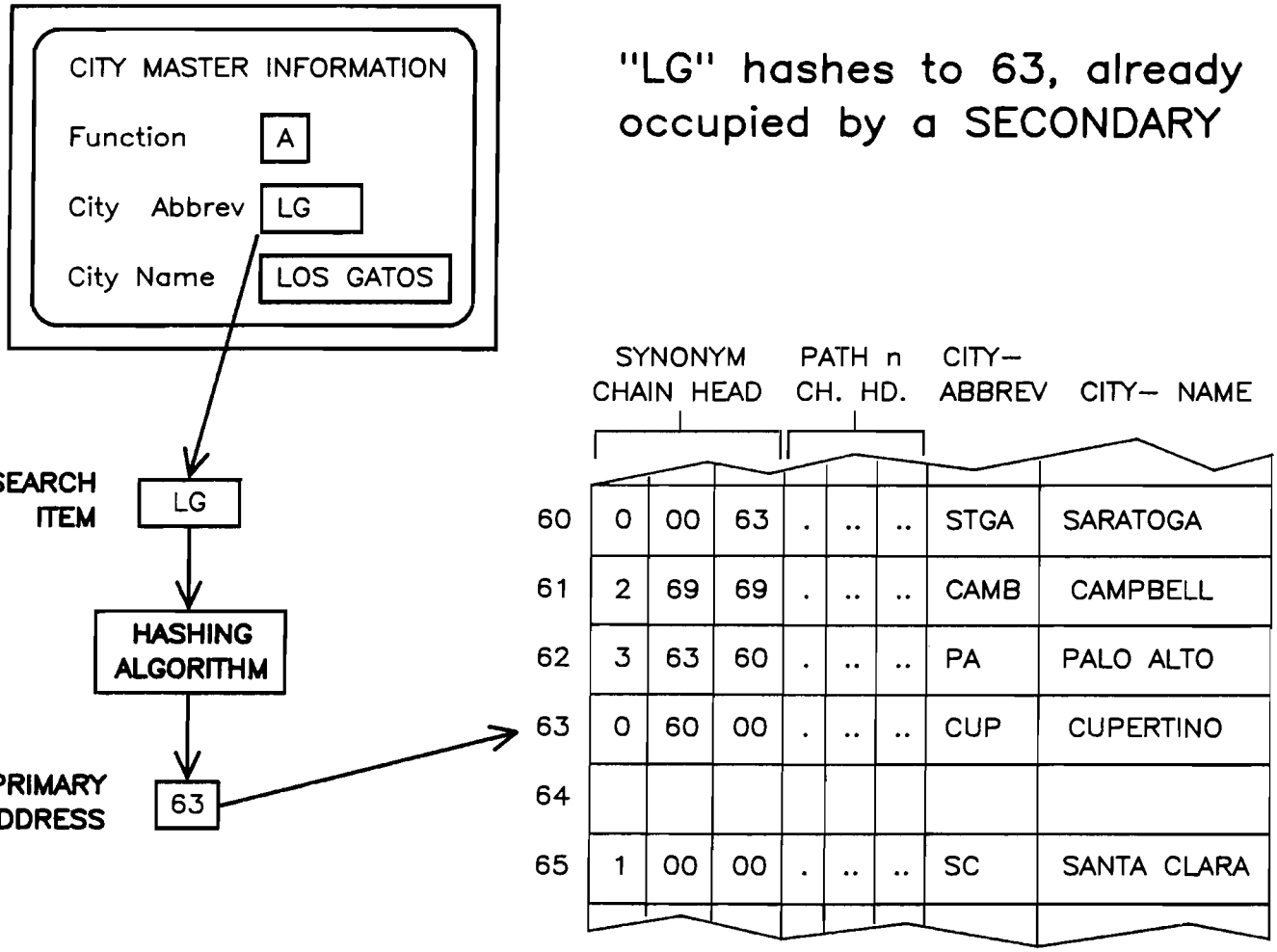
* Occurs when a new master entry's search item value hashes to an address already occupied by a SECONDARY

OR

when the PRIMARY in a synonym chain is deleted

notes:

references:

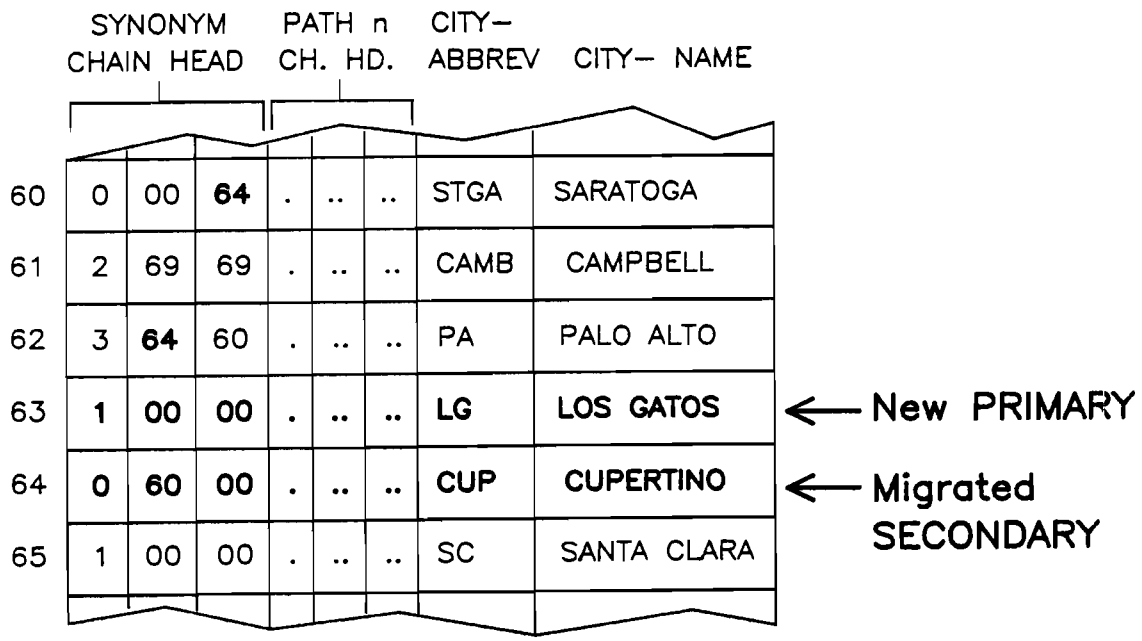


"LG" hashes to 63, already occupied by a SECONDARY

notes:

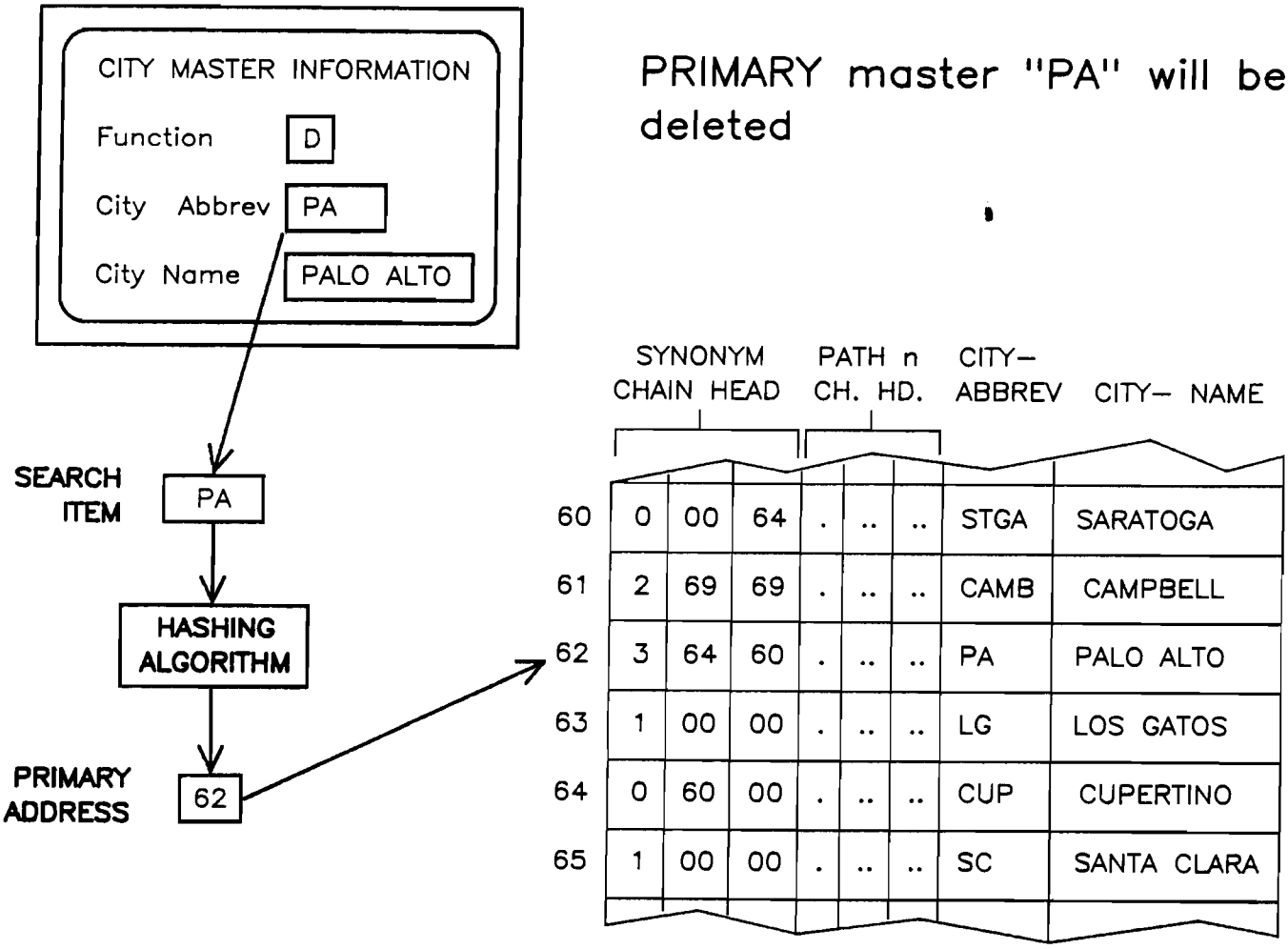
references:

"CUP" migrates to 64



notes:

references:

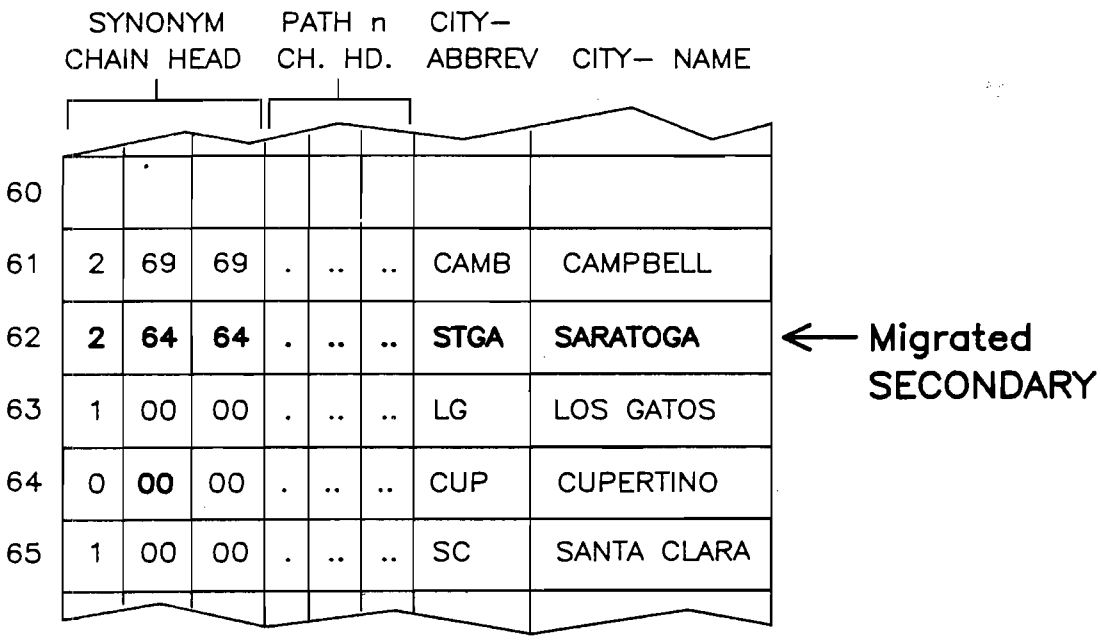


PRIMARY master "PA" will be deleted

notes:

references:

SECONDARY "STGA" moves to PRIMARY position



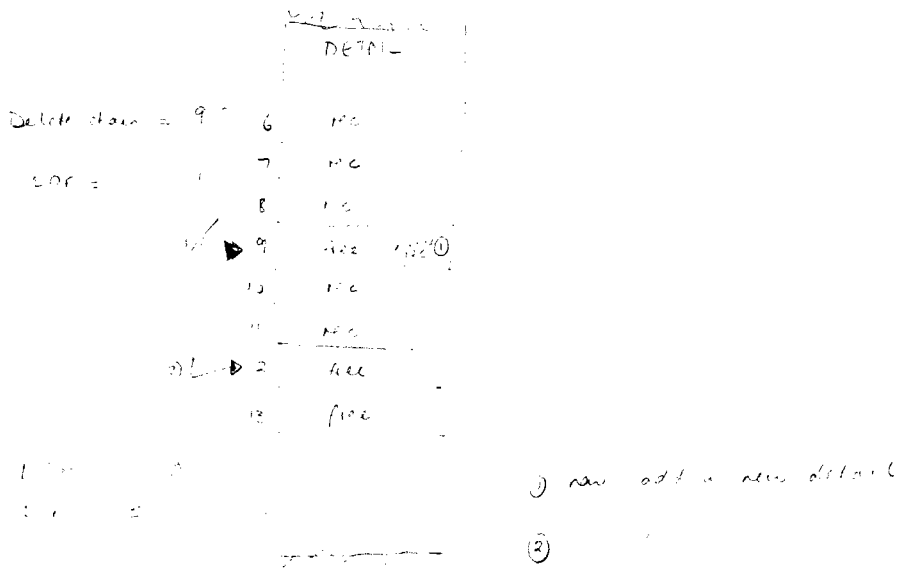
notes:

references:

DETAIL RECORD PLACEMENT

* Placement is controlled by two pointers
  - Delete Chain Pointer – entry most recently deleted
  - End–of–file Pointer – highest relative record address used so far in this set
* If the Delete Chain Pointer is zero, the End–of–file Pointer is used to place details

notes:



references:

CHAIN MAINTENANCE

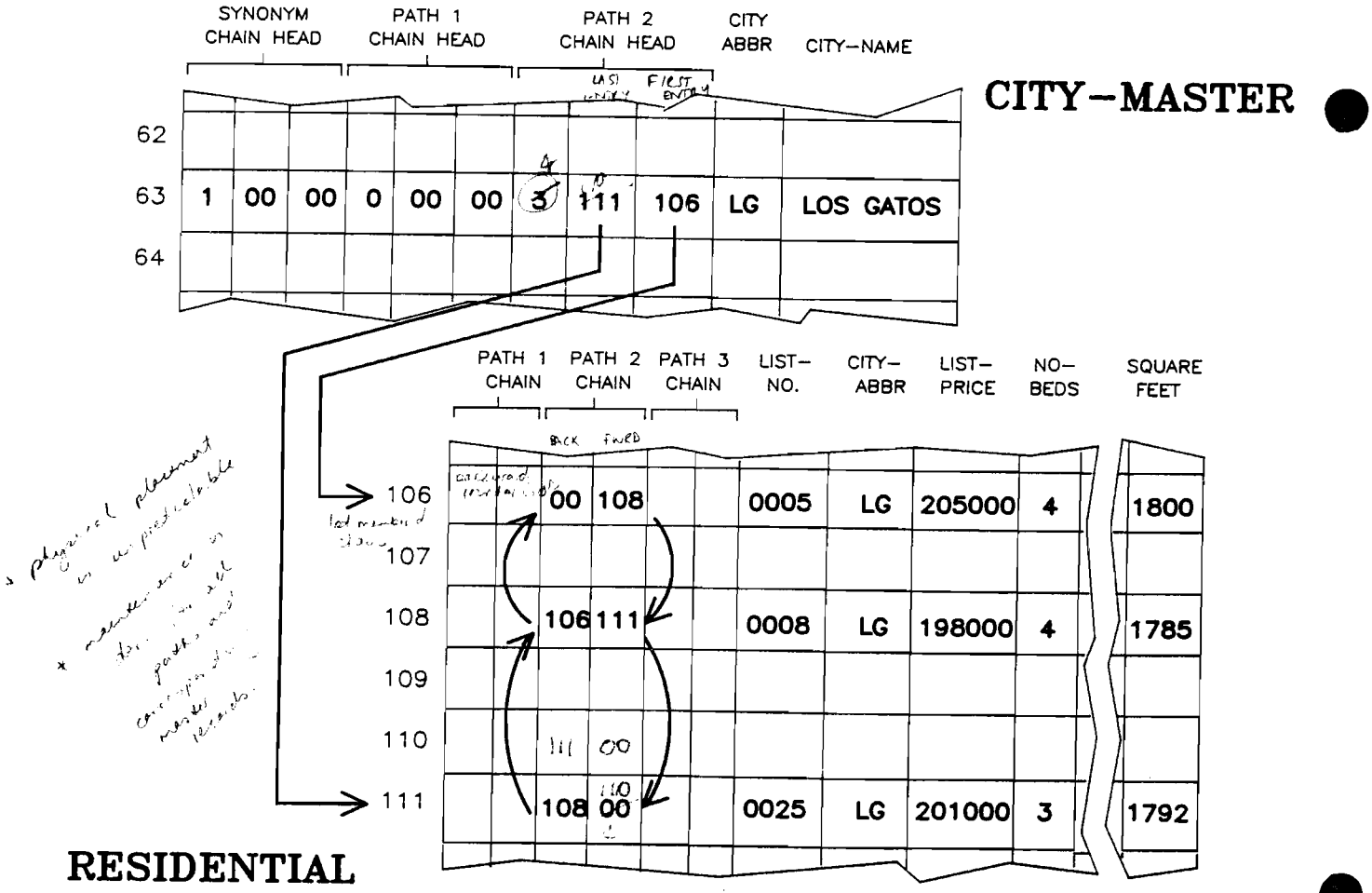
* New detail records are added to the "end" of the logical chain — chronological order

* Pointers must be updated in the old end of chain and the master record's chain head

* Count must be incremented in the master chain head

notes: LOGICAL placement — key points are changed to reflect the new order

references:



notes:

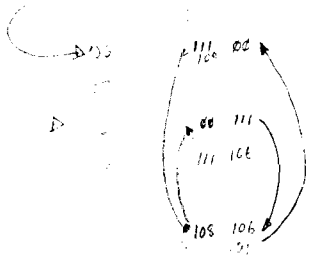
references:

SORTED CHAIN MAINTENANCE

* IMAGE reads existing chain backwards until it finds the proper logical spot

* If sort field values match, IMAGE uses fields following the sort field to determine order (extended sort field)

notes:



references:

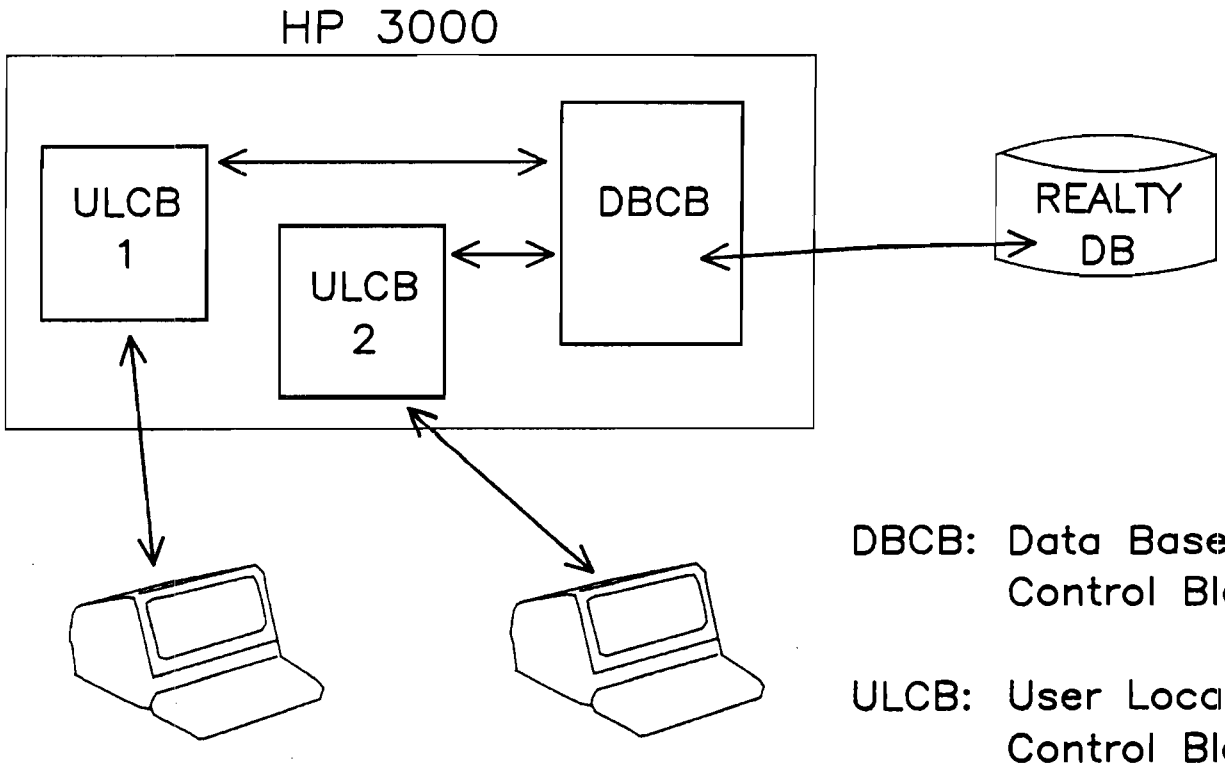
IMAGE

CONTROL BLOCKS

notes:

references:

CONTROL BLOCKS



DBCB: Data Base Control Block

ULCB: User Local Control Block

(XDS)

HEWLETT PACKARD

notes:

references: See IMAGE Reference Manual, Section X to calculate control block sizes

DATA BASE CONTROL BLOCK (DBCB)

* 1 per open data base
* Shared by all users of that data base
* Contains global root file information, pointers, buffers
* Created when first user DBOPEN's data base

notes:

DBUTIL
>SHOW

references:

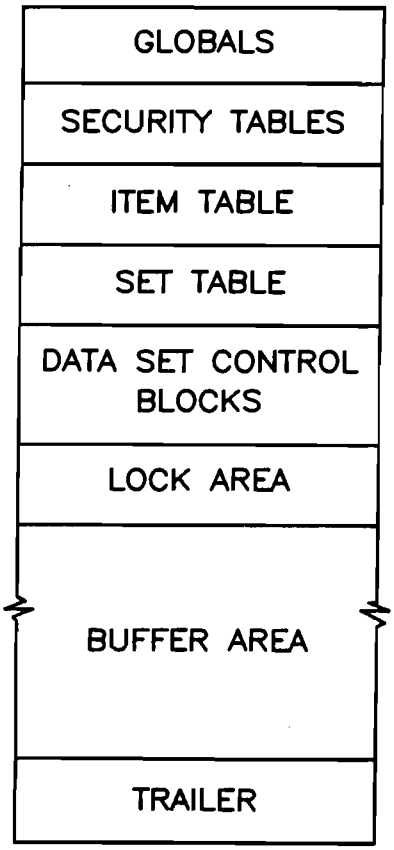
USER LOCAL CONTROL BLOCK (ULCB)

* 1 for every DBOPEN to each data base used
* Contains information local to this access path to data base such as current path, current record, user's security table and last procedure called

notes:

references:

BUFFERS AND THE DBCB



* Buffers are shared by all users of this data base
* Each buffer is largest data set block size plus 10 words
* Number of buffers can be modified using DBUTIL SET

notes:

references:

BUFFERS

* Default buffer allocation at DBOPEN:

  Max paths into any set + 3
  or
  8 buffers
  whichever is greater

* 1 buffer added for every 2 additional users

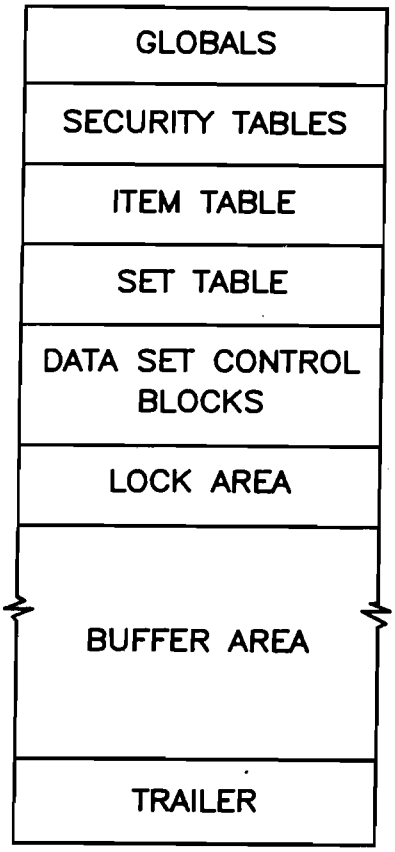
* DBCB expands and contracts as users DBOPEN and DBCLOSE

notes:

- minimum buffers = 4; maximum = 255

references:

LOCK AREA AND THE DBCB



* Keeps track of all users' DBOPEN's by pin and open count
* Records base, set and entry level lock requests
* Requests are chained together as a function of time

notes:

references:

INTERNALS

OF

DATA BASE PROCEDURES

notes:

references:

DATA BASE PROCEDURES
GENERAL

* Data set and item names are converted to numbers by IMAGE before calls are executed

* Output buffers are assembled in the "TRAILER" of the DBCB

* Remember, overhead for a transaction is the accumulation of IMAGE overhead for each and every procedure call

notes:

references:

DATA BASE PHYSICAL INTEGRITY

To maintain data base physical integrity:

* System aborts and user breaks are disabled during IMAGE procedures
* DBCB is locked by one procedure at a time; other calls queue up
* All changed buffers are posted before exiting the procedure

notes:

references:

DBOPEN

1. Open root file
2. Create DBCB (if this is first user)
3. Create ULCB
4. Determine security restrictions, access mode, concurrent modes allowed
5. Do buffer expansion if indicated

notes:

references:

DBCLOSE

1. If mode 2, close this data set and exit
2. If mode 3, set ULCB pointers for this data set file to first and last records and exit
3. Close all open data sets
4. Close root file
5. Delete ULCB
6. Delete DBCB (if this is last user)
7. Do buffer contraction if indicated

notes:

references:

DBFIND

1. Open detail set if not already open
2. Determine path to master
3. Open master set if not already open
4. Hash search item value
5. Read desired master block into DBCB buffer
6. If search item matches, go to 8
7. If search item does not match, save forward synonym pointer and go to 5
8. Set up first and last entry pointers to detail data set in ULCB

notes:

references:

DBGET MODE 2
(serial)

1. Open set if not already open
2. Verify LIST parm and read capability
3. Advance current record pointer by 1
4. If record needed is not in buffer, read in desired block
5. If bit map bit is "1", go to 7
6. If bit map bit is "0", go to 3
7. Set backward and forward chain pointers in ULCB
8. Move requested data items to stack

notes:

references:

DBGET MODE 4
(directed)

1. Open set if not already open
2. Verify LIST parm and read capability
3. Set current record pointer to ARGUMENT
4. If record needed is not in buffer, read in desired block
5. Verify that bit map bit is "1"
6. Set backward and forward chain pointers in ULCB
7. Move requested items to stack

notes:

references:

DBGET MODE 5
(chained)

1. Open set if not already open
2. Verify LIST parm and read capability
3. Set current record pointer to forward pointer
4. If record needed is not in buffer, read in desired block
5. Verify that bit map bit is "1"
6. Set backward and forward chain pointers in ULCB
7. Move requested items to stack

notes:

references:

DBGET MODE 7
(calculated)

1. Open set if not already open
2. Verify LIST parm and read capability
3. Hash search item value to get primary address
4. If record needed is not in buffer, read in desired block
5. If ARGUMENT parm matches this master's search item value, go to 7
6. If ARGUMENT parm does not match, set current record to forward synonym pointer and go to 4
7. Set backward and forward chain pointers
8. Move requested items to stack

notes:

references:

DBUPDATE

1. Check for update access

2. Check read access for items in LIST parm

3. If record to be updated still not in buffer, read desired block into buffer

4. Compare new value of each item in LIST to old value; if changed, check item security and move value to buffer

5. Write buffer to disc

notes:

references:

DBPUT (MASTER)

1. Open set if not already open
2. Validate LIST parm
3. Check set capacity
4. Hash search item
5. Read desired block into DBCB buffer
6. If primary adress not already occupied:
   - copy new data to buffer
   - set bit map bit to "1"
   - decrement free space counter
   - set pointers and counts to 0
   - set synonym count to 1
   - write buffer to disc and exit

notes:

references:

DBPUT (MASTER)

7. If primary address occupied by another primary:

   - find end of synonym chain by using forward synonym pointer to read each block until pointer is 0*
   - search bit map of last synonym's block for nearest free record
   - update last synonym's pointers and write its buffer to disc
   - update primary's count and last/first entry pointers if necessary, and write its buffer to disc
   - rest of processing same as 6
   - exit

notes:

references:

• DBPUT (MASTER)

8. If primary address occupied by a secondary:
   - search bit map for nearest free record
   - move secondary to this record
   - read in secondary's forward and backward members, update pointers and write buffers to disc
   - using original primary address, do rest of processing as in 6
   - exit

notes:

references:

DBPUT (DETAIL)

1. Open set if not already open
2. Validate LIST parm
3. Check set capacity
4. If delete chain pointer is non–zero, use it to add this record; if it is zero, use the end–of–file pointer
5. Set bit map bit to "1"
6. For each path:
   - determine associated master
   - hash search item
   - read in desired master block
   - compare search item values; if not equal, perform `synonym chase' until equal
   - modify backward pointer and chain count

notes:

references:

DBPUT (DETAIL)

7. For each chain:
   - modify previous record forward pointer
   - set forward and backward pointers of new entry

8. Copy new data to buffer

9. Decrement free space counter

10. Update End-of-file or Delete Chain pointer

11. Write buffer to disc

notes:

- If sorted chain, read chain backwards to determine location.

references:

DBDELETE (MASTER)

1. If record to be deleted not still in buffer, read desired block into buffer

2. Check for count of 0 in all detail chain heads

3. If this is a primary path with no secondaries:
   - set record contents to binary zeroes
   - set bit map bit to "0"
   - update free space counter
   - write buffer to disc and exit

notes:

references:

DBDELETE (MASTER)

4. If primary with secondaries:
   - using forward synonym pointer, read first secondary's block into buffer
   - move this secondary to primary's record location
   - update all pointers and counts
   - read next secondary into buffer, update its pointers and write to disc
   - set first secondary's contents to binary zeroes, set its bit to "0", write buffer to disc

notes:

references:

DBDELETE (MASTER)

5. If this is a secondary:
   - read into buffer, the synonyms before and after this secondary in the synonym chain
   - update pointers and write buffers to disc
   - read primary into DBCB buffer
   - decrement synonym count and, possibly, last/first entry
   - perform rest of processing as in 3 and exit

notes:

references:

DBDELETE (DETAIL)

1. If record to be deleted not still in buffer, read desired block into buffer
2. For next path:
   - determine associated master
   - hash search item
   - read in desired master block
   - compare search item values; if not equal, perform `synonym chase' until equal
   - modify first entry pointer, if this is first record in detail chain; last entry pointer, if last record
   - decrement count (if this and all other chain counts are 0 and this is automatic master, schedule master for deletion)
   - write buffer to disc
   - if there are more paths, go to 2

notes:

references:

DBDELETE (DETAIL)

3. For next detail chain:
   - read in predecessor's block
   - update forward pointer
   - read in successor's block
   - update backward pointer
   - write buffer to disc
   - if there are more chains, go to 3
4. Set record contents to binary zero
5. Set bit map bit to "0"
6. Update delete chain head pointer
7. Write buffer to disc

notes:

references:

PERFORMANCE

CONSIDERATIONS

notes:

references:

TOPICS

* CONSIDERATIONS

* EFFICIENT PROGRAMMING TECHNIQUES

notes:

references:

What resources affect performance?

* CPU time
* Main Memory
* I/O

notes:

references:

GOAL

Minimize use of CPU, Memory, and I/O resources while meeting the needs of the end—user.

notes:

references:

ITEMS, SETS, & PATHS

more items
more sets
more paths

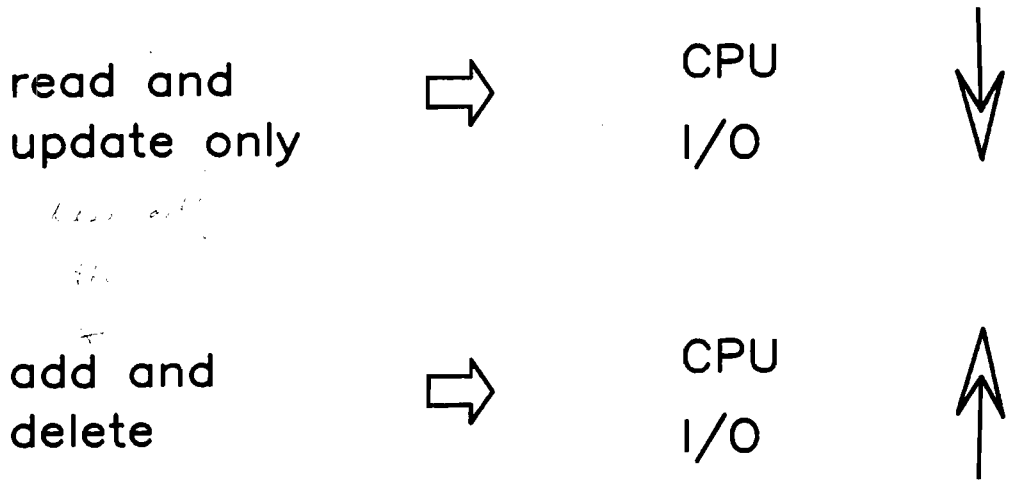
⇨ increasing complexity ⇨

CPU
MEMORY
I/O

notes:

references:

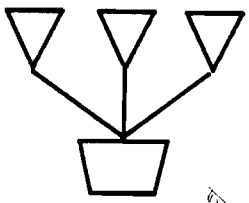
DATA BASE APPLICATION TYPE



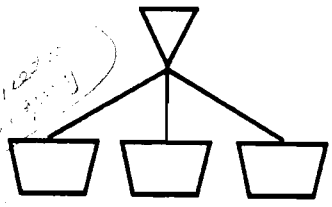
notes:

references:

PATHS



for adds and deletes ⇨ I/O ⇧



for adds and deletes ⇨ I/O ⇩

notes:

references:

SYNONYM CHAINS

byte-type keys

capacity = prime #

20-30% free space

⇨ fewer synonyms

shorter synonym chains

⇨ I/O ⇩

HEWLETT PACKARD

notes:

- byte-types are X, U, Z, or P

references:

DETAIL CHAINS

many adds and deletes ⇨ increased disorganization ⇨ for chained reads I/O ↑

select PRIMARY PATH = most accessed chain ⇨ regular DBUNLOAD and DBLOAD ⇨ for chained reads I/O ↓

notes:

```
Schema example -

  NAME:       RESIDENTIAL, DETAIL  (10,20/30);
  ENTRY:      LISTING-NR      (LISTNR-MASTER),
              CITY-ABBR       (!CITY-MASTER),
              LIST-PRICE      (LIST-PRICE-MSTR (SQUARE-FEET)),
              NUMBER-BEDS     (BEDS-MASTER),
              .
              .
              .
```

references:

SORTED CHAINS

many sorted chains ⇨ for adds ⇨ I/O ↑

short chains

pre-sorted transactions ⇨ for adds ⇨ I/O ↓

sort field at EOR

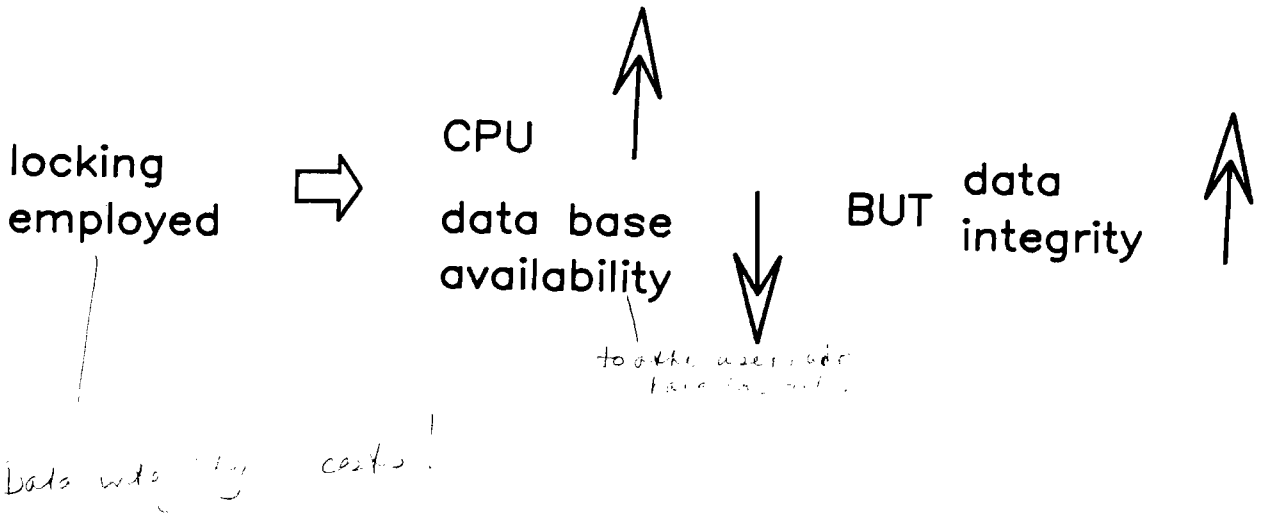
notes:

Schema example -

```
NAME:       COMMERCIAL, DETAIL  (10,20/30);
ENTRY:      LISTING-NR     (LISTNR-MASTER),
            ZONING-CODE    (!ZONING-MASTER),
            LIST-PRICE     (LIST-PRICE-MSTR (SQUARE-FEET)),
            CITY-ABBR      (CITY-MASTER),
            STREET-ADDRESS,
            .
            .
            .
            SQUARE-FEET;
CAPACITY:   300;
```

references:

LOCKING



notes:

references:

LOCKING LEVELS

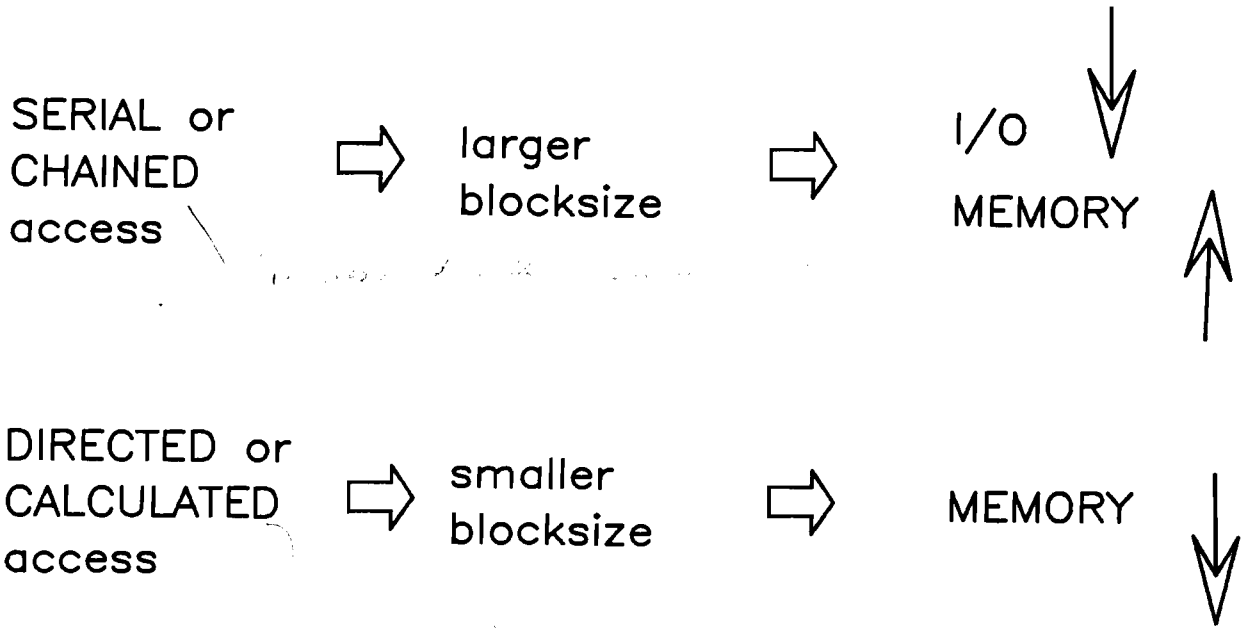
lowest possible locking level ⇨ programming complexity ↑ BUT data base availability ↑

entry–level locking AND common lock item, pre–sorted descriptor ⇨ data base availability ↑

notes:

references:

BLOCKSIZE



notes:

- set blocksize by using $CONTROL BLOCKMAX=
  minimum=128W; maximum=2048W;default=512W

references:

BUFFERS

default buffers with frequent DBOPEN's and DBCLOSE's ⇨ DBCB expansion/ contraction ⇨ I/O ⇑

stable buffer allocation for range of users ⇨ reduced DBCB expansion/ contraction ⇨ I/O ⇓ MEMORY ⇑

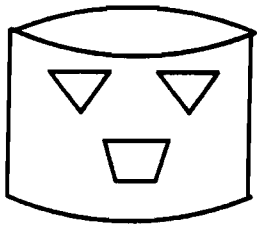
notes:

- to allocate 12 buffers for up to 10 users:

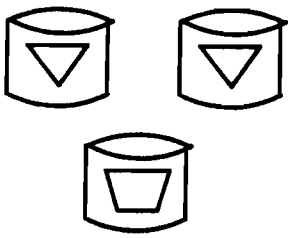
```
:RUN DBUTIL.PUB.SYS
>>SHOW REALTY BUFFSPECS FOR DATABASE REALTY

BUFFER SPECIFICATIONS:
8(1/2),9(3/4),10(5/6),11(7/8),12(9/10),13(11/12)...

>>SET REALTY BUFFSPECS=12(1/10),13(11/12)...
>>EXIT
```

references IMAGE Reference Manual, Section VIII, Using the Data Base Utilities

FILE PLACEMENT



⇨ increased head contention ⇨ wait for I/O ↑



⇨ reduced head contention ⇨ wait for I/O ↓

notes:

- Use ":RESTORE" with the ";DEV=" parameter to direct files to a particular disc.

references:

MULTIPLE DATA BASES

more data bases ⇨ increased complexity ⇨ CPU ↑, MEMORY ↑, I/O ↑

HOWEVER...

more data bases ⇨ flexibility ↑, contention ↓, recovery time ↓

notes:

references:

BACKUP AND RECOVERY

frequent backups ⇨ data base is often unavailable — BUT — recovery is easier and faster

loss of data is reduced

IMAGE logging ⇨ CPU ↑ MEMORY ↑ I/O ↑ — BUT — loss of data ↓ data base is available ↓ recovery time ↓

notes:

references:

EFFICIENT

PROGRAMMING

TECHNIQUES

notes:

references:

DBOPEN

* Use open mode that protects well, but allows as many concurrent users as possible

DBCLOSE

* Use MODE 2 to close data set if you will never access it again

* Use MODE 3 to reset pointers to 0 and avoid overhead of other modes

GENERAL RULE:

Minimize the number of DBOPEN's to reduce use of all resources!!

notes:

references:

DBGET

* LIST = @ reduces item-by-item checking and assembly of buffer

* LIST = * on second and later calls avoids list processing and security check

* Pass numbers instead of names in LIST and SET to avoid Item and Set Table searches

* Use partial LIST to retrieve few items from a large record to avoid security check.

notes:

- Use DBINFO procedure calls to retrieve data item and data set numbers. This increases data independence.

references:

DBUPDATE

* Use partial LIST to reduce security check
* Use LIST = @ only if changing most non-key items or if changes are unpredictable
* Ensure key is unchanged
* Ensure write access to all items that may be changed
* Write access at set level is faster

notes:

references:

DBPUT :

* LIST = @ reduces item-by-item buffer assembly

* LIST = * on second and later calls avoids list processing and security check

* Ensure all keys are present

REMEMBER....

DBPUT's and DBDELETE's to MASTER sets may cause migrating secondaries!

notes:

references:

LAB 12

PERFORMANCE CONSIDERATIONS

notes:

references:

Please answer the following questions. Try to answer them without looking at your notes. If you have trouble, turn back to the pages in this section to get help with the answers.

Please answer TRUE or FALSE or ? (I don't know)

1. A large blocksize is the best choice for a data set that is usually accessed in serial mode (DBGET mode 2 or 3).

2. The entire data base should be kept on one disc drive in order to reduce head contention.

3. The capacity for any set should only be great enough to hold the current records so that disc space is not wasted.

4. The number of synonyms in a master set can usually be reduced by choosing a prime number capacity for that set.

5. When many add (DBPUT) and delete (DBDELETE) transactions are required for a detail data set, performance can be improved by increasing the number of paths it has to master sets.

6. In general, more items, sets, and paths in your data base means more system resources are required to run your application.

7. When you design a data base application to perform well on the 3000, your only goal should be to satisfy every request of the end-users.

8. One advantage of data base logging is that it does not use any system resources.

9. A "*" in the LIST parameter of an IMAGE procedure call can be used to minimize overhead.

10. An application using data entry locking should have better performance than the same application using data set locking, no matter which data item each program of the former uses in its lock descriptor.

DESIGN

CONSIDERATIONS

notes:

references:

TOPICS

* CONSIDERATIONS
* DESIGN APPROACHES
* COMMON DESIGN DECISIONS

notes:

references:

DESIGNER'S JOB

* Define application and data base design

* Define and prioritize global concerns for the application

* Consider trade—offs

notes:

references:

DESIGN ACTIVITIES

* APPLICATION DESIGN

  Logical design which defines current and future information requirements, functions to be performed, data elements, security, keys, relationships.

* DATA BASE DESIGN

  Physical design which provides structures and access methods to implement the application, performance, backup and recovery, security.

notes:

references:

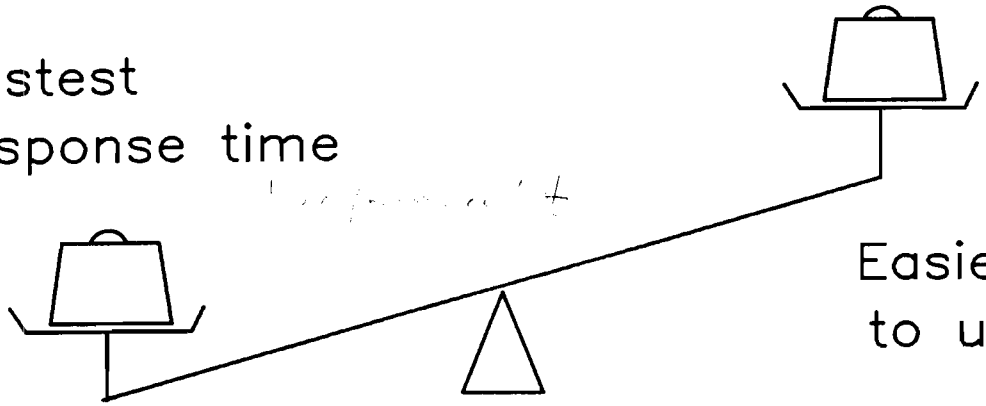
TYPICAL DESIGN CONCERNS

* Flexibility and complexity of end—user needs
* Security
* Performance
* Development time
* Data independence
* Disc storage
* Maintenance time
* Backup and recovery time

notes:

references:

CONSIDER PRIORITIES!



CONSIDER TRADE–OFFS!

HEWLETT PACKARD

notes:

references:

DESIGN

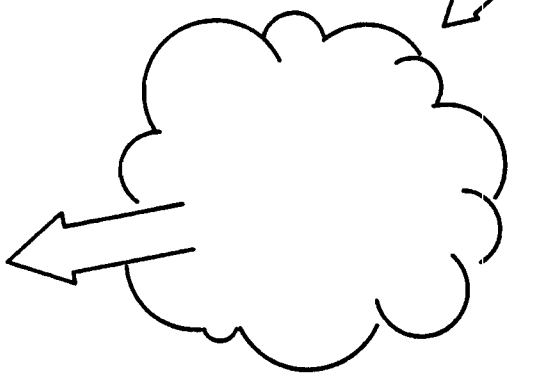
APPROACHES

notes:

references:

INTUITIVE DESIGN APPROACH

What data is needed? ⇨ What are the relationships between data? ⇨ How is data retrieved/ what are keys? ⇨ What are fast access keys?



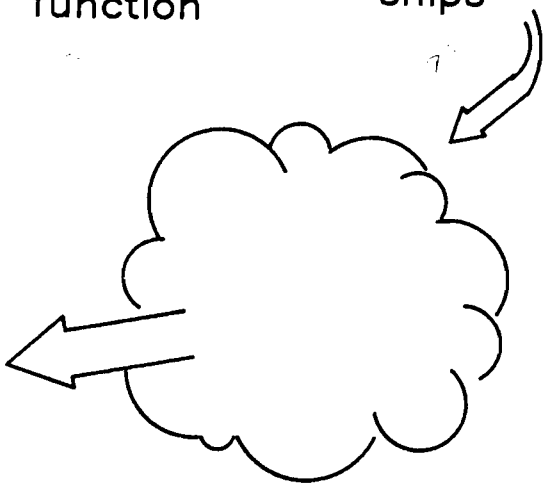
Draw the data base and optimize

notes:

references:

FUNCTIONAL APPROACH

What functions are to be performed? ⇨ What information is needed for each function? ⇨ Group information by function ⇨ Identify relation-ships



Draw the data base and optimize

notes:

references:

WONDER REALTY

Residential Listing Application:

1. Identify each residence
2. Identify each residence by city
3. Identify each residence by price
4. Identify each residence by number of bedrooms
5. Identify each residence by number of bathrooms
6. Identify each residence by unique key
7. Identify each city

notes:

references:

FUNCTION MATRIX

	Function Number:		1	2	3	4	5	6	7
	Function Description:		Identify each house	Identify by city	Identify by price	Identify by # beds	Identify by # baths	Identify by listing#	Identify each city
	Data Item	Item No.	DATA ITEM FUNCTIONS						
*	# bedrooms	1	X			X			
*	# bathrooms	2	X				X		
	square feet	3	X						
	dining rm	4	X						
	sold	5	X						
*	price	6	X		X				
	owner	7	X						
	street addr.	8	X						
	city	9							X
	zip	10	X						
	phone	11	X						
*	listing #	12	X	X	X	X	X	X	
*	city abbrev.	13	X	X					X

* = key

notes:

references: See APPENDIX G for a blank Function Matrix form

RELATION MATRIX

	Data Item	Item No.	1	2	3	4	5	6	7	8	9	10	11	12
*	# bedrooms	1												
*	# bathrooms	2	N<->M											
	square feet	3	1 ->N	1 ->N										
	dining room	4	1 ->N	1 ->N	1:1									
	sold	5	1 ->N	1 ->N	1:1	1:1								
*	price	6	N<->M	N<->M	1 ->N	1 ->N	1 ->N							
	owner	7	1 ->N	1 ->N	1:1	1:1	1:1	1 ->N						
	street addr	8	1 ->N	1 ->N	1:1	1:1	1:1	1 ->N	1:1					
	city	9	X	X	X	X	X	X	X	X				
	zip	10	1 ->N	1 ->N	1:1	1:1	1:1	1 ->N	1:1	1:1	X			
	phone	11	1 ->N	1 ->N	1:1	1:1	1:1	1 ->N	1:1	1:1	X	1:1		
*	listing #	12	1 ->N	1 ->N	1:1	1 ->N	1:1	1 ->N	1:1	1:1	X	1 ->N	1:1	
*	city abbrev	13	N<->M	N<->M	1 ->N	1 ->N	1 ->N	N<->M	1 ->N	1 ->N	1:1	1 ->N	1 ->N	1 ->N

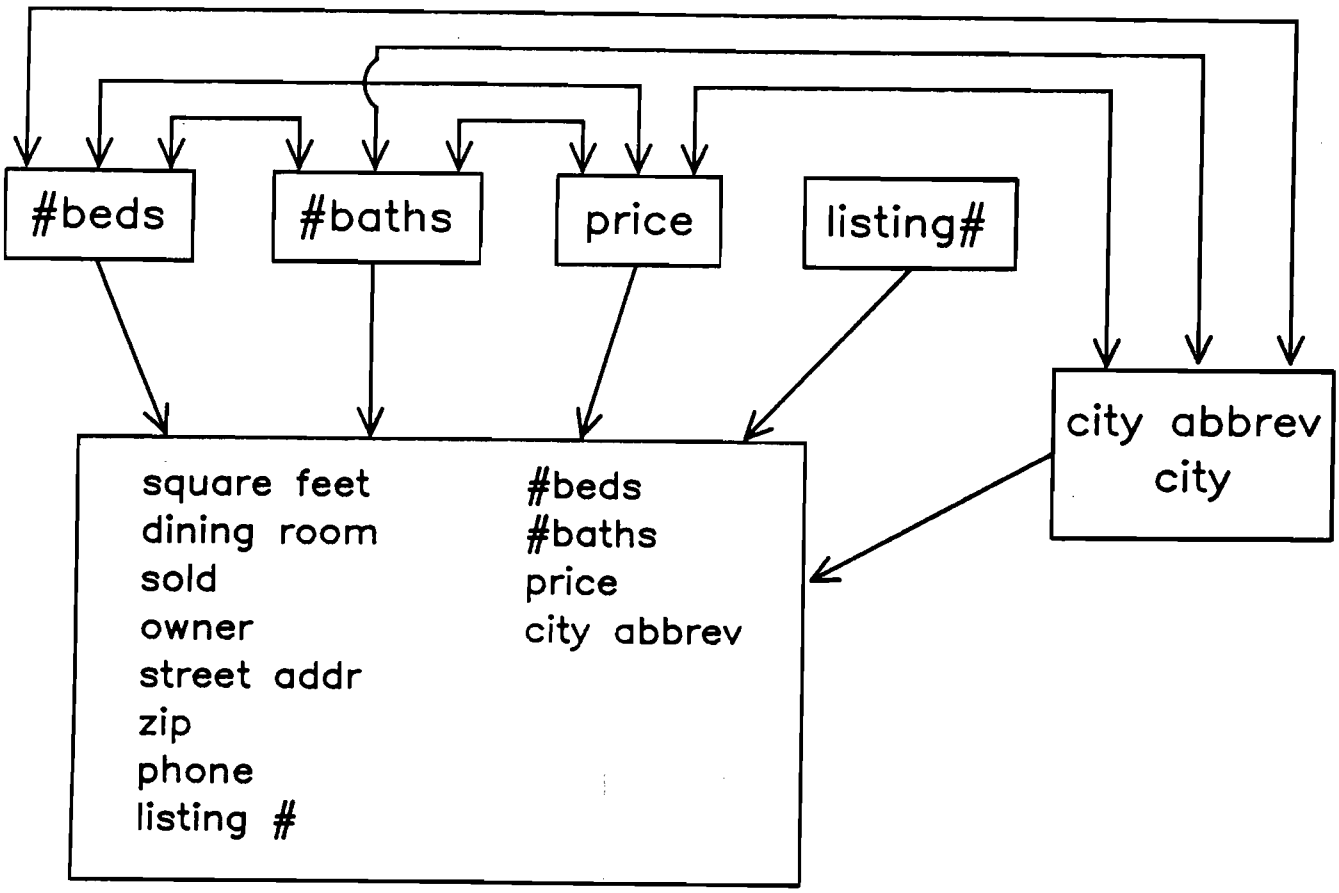
* = key

notes:

1:1 one to one
1->N one to many
N<->M many to many
X no relationship

references: See APPENDIX G for a blank Relation Matrix form

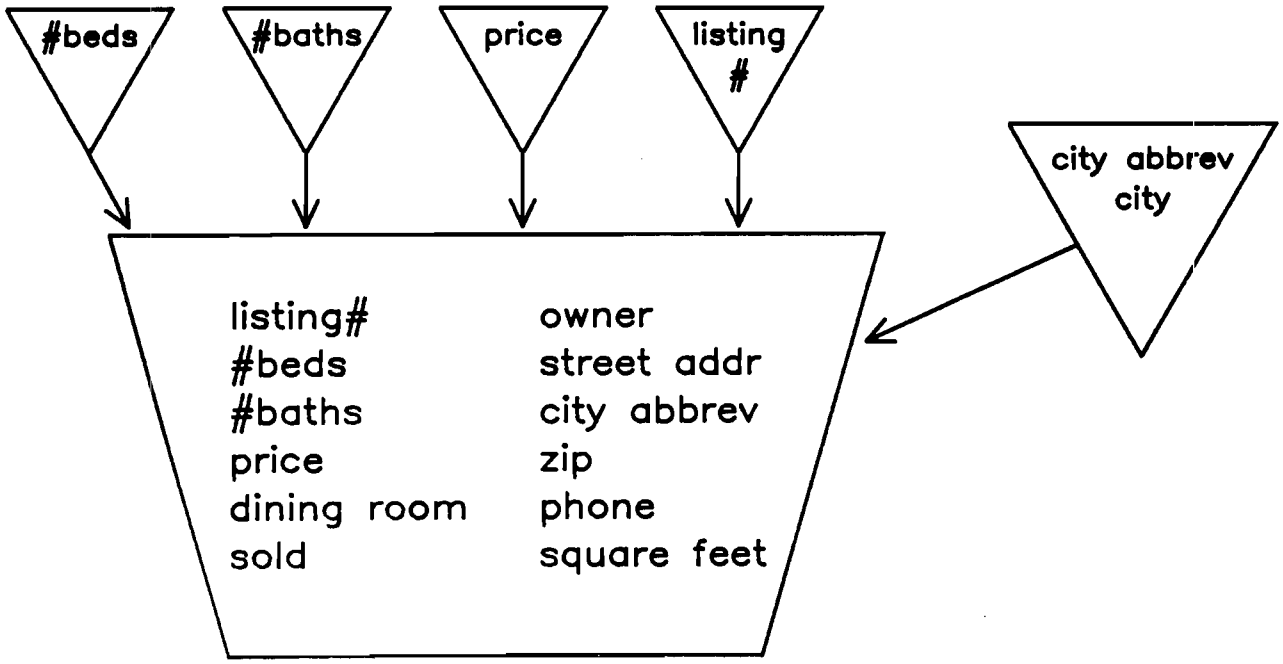
REALTY – First Pass



notes:

references:

REALTY – Second Pass



notes:

references:

COMMON

DESIGN

DECISIONS

notes:

references:

KEY SELECTION

* How do users look up data?

* What other keys are needed for fast or frequent access?

Considerations:

How much?
How fast?
How often?

Performance impact

Example: LIST–PRICE and LISTING–NR keys

HEWLETT PACKARD

notes:

references:

SETS AND PATHS

* How is each data item related to every other data item?

* What does it take to describe an entity or to accomplish a function?

  Considerations:

  1 ⟷ 1 relationships
  1 → many relationships
  many ⟷ many relationships
  no relationship

Example: RESIDENTIAL detail set

notes:

references:

MASTER vs. DETAIL SET

* Detail sets usually store multiple occurrences of information

* Master sets usually hold unique values (keys) for fast retrieval

* Masters may contain the information that describes the key

Example: RESIDENTIAL and LISTNR–MASTER sets

notes:

references:

MANUAL vs. AUTOMATIC

* Are key values numerous and unpredictable? (automatic)

* Must key values be valid? (manual)

* Must additional data be stored with the key? (manual)

Example: LIST–PRICE–MSTR and CITY–MASTER sets

notes:

references:

CAPACITY

* What is the current capacity required to store all records?

* How much and how fast will this figure grow?

* Set detail set capacity to current need plus reasonable growth.

* Set master set capacity to current need plus growth plus 20–25% free space.

Considerations:

Disc space
Hashing
Time to back up

notes:

references:

SINGLE vs. MULTIPLE USERS

* Must any function have exclusive access to the data base?

* Must readers be protected from changes during their transactions?

* Must updaters be protected from other updaters?

Considerations:

Performance impact
Open modes
Locking levels

notes:

references:

QUERY FOR PRODUCTION

* How often will this report be executed and against which other functions?
* Does the function require an audit trail or validation function?

Considerations:

Performance impact
Time and $ to develop report in another language

notes:

references:

SINGLE vs. MULTIPLE DATA BASES

* Are IMAGE item or set limits reducing application flexibility?

* Are data logically categorized by function with little overlap?

Considerations:

Complexity of programming
Memory usage
Backup and recovery

notes:

references:

INTERACTIVE vs. BATCH

* Is this primarily a reporting or update application?

* How current must information be?

* Must changes to key values be made on-line?

Considerations:

Users' need for current information
Performance impact

notes:

references:

BACKUP and RECOVERY

* How critical is this data to the operation of the business?

* How often must the data be available to users? (hours per day, days per week)

* How much data can we afford to lose in case of a catastrophic failure?

* How long can we be down for recovery?

Considerations:

Time to backup
Performance impact of logging

notes:

references:

MISCELLANEOUS

TOPICS

notes:

references:

REMOTE

DATA BASE ACCESS

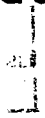
(RDBA)

notes:

references:

REMOTE DATA BASE ACCESS (RDBA)

Wonder Realty has become very successful! They now have offices across the U.S., each with its own HP3000 running our realty application.



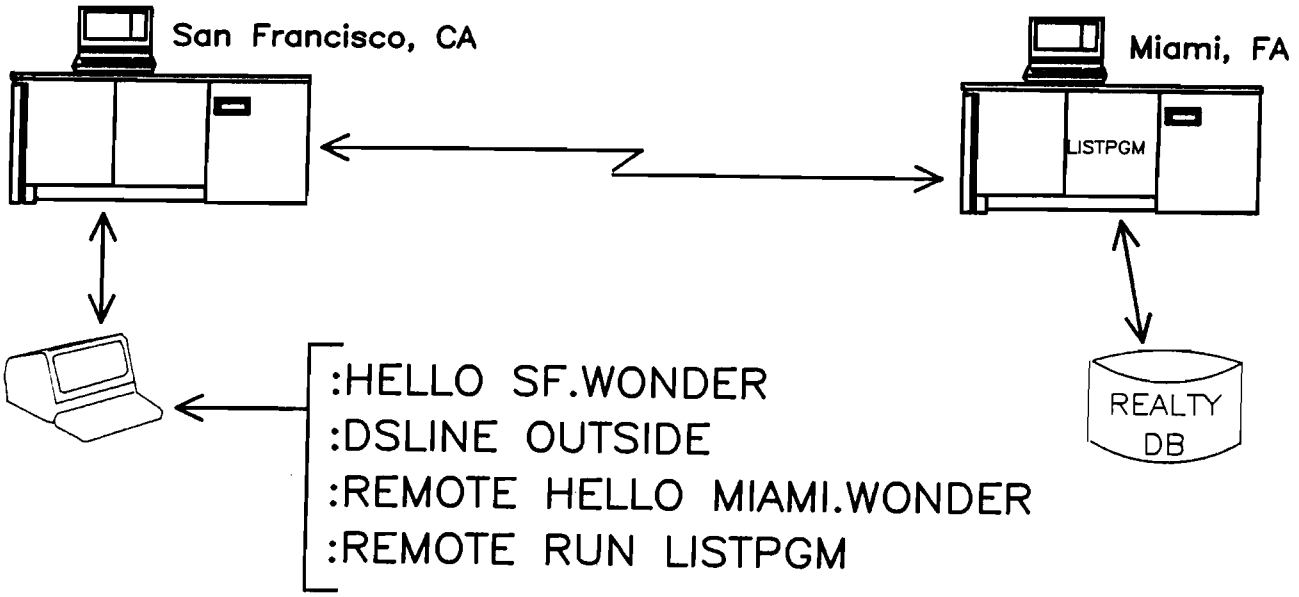
PROBLEM: How can agents in California check the latest listings in Florida in order to help their clients relocate?

notes:

references:

SOLUTION: Distributed Systems (DSN/DS) and Remote Data Base Access (RDBA)

RDBA METHOD #1 – REMOTE COMMANDS



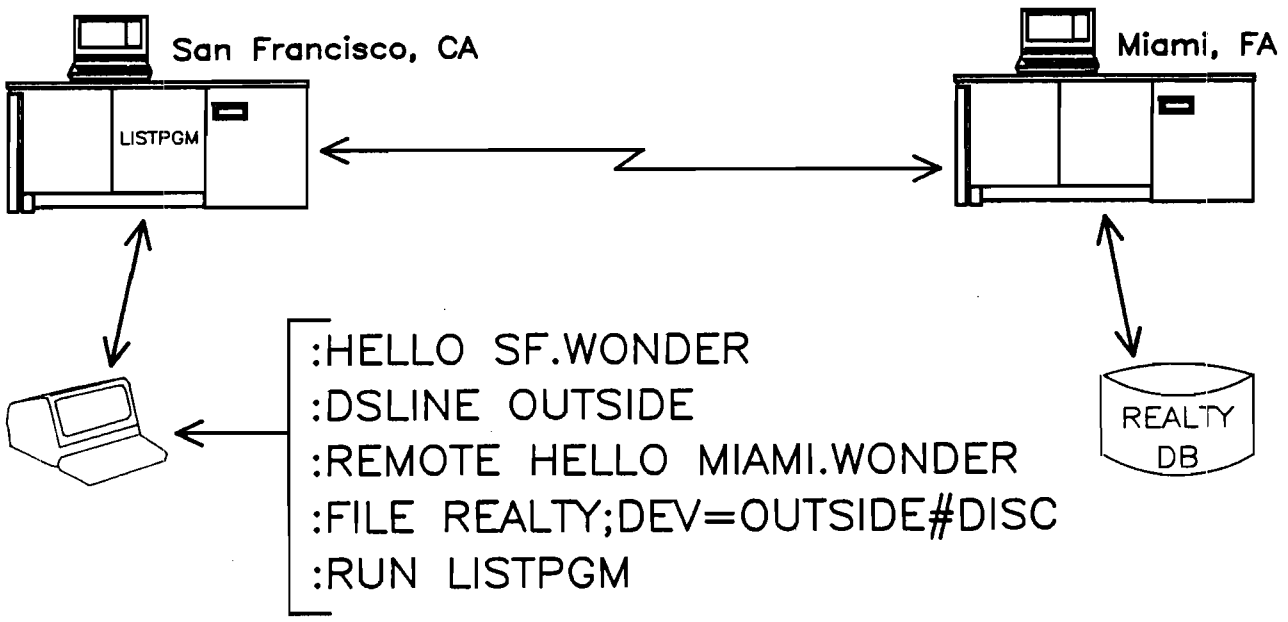
notes:

"OUTSIDE" is the device class name for the DS line on the San Francisco HP3000. We could have used the logical device number.

You can run QUERY on the remote machine by substituting ":REMOTE RUN QUERY.PUB.SYS" for the fourth command.

references DS/3000 Reference Manual
IMAGE Reference Manual, Section IX, Using a Remote Data Base

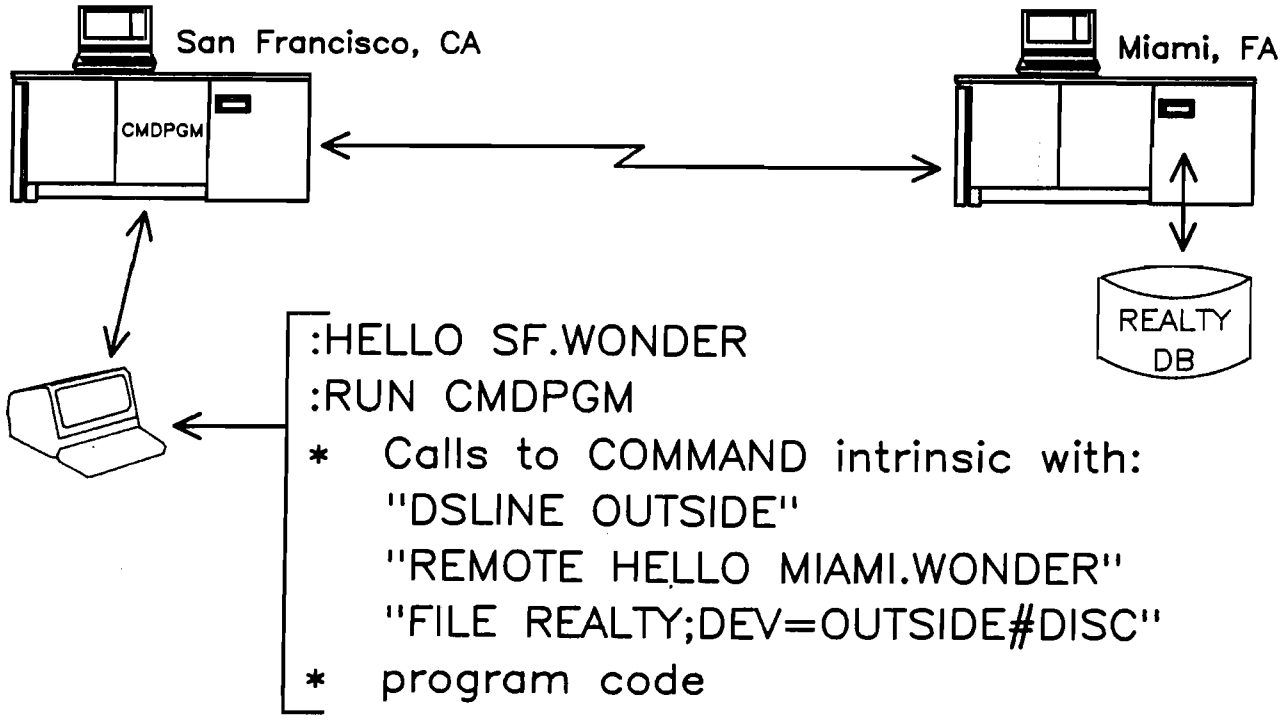
RDBA METHOD #2
REMOTE FILE ACCESS



notes:

references:

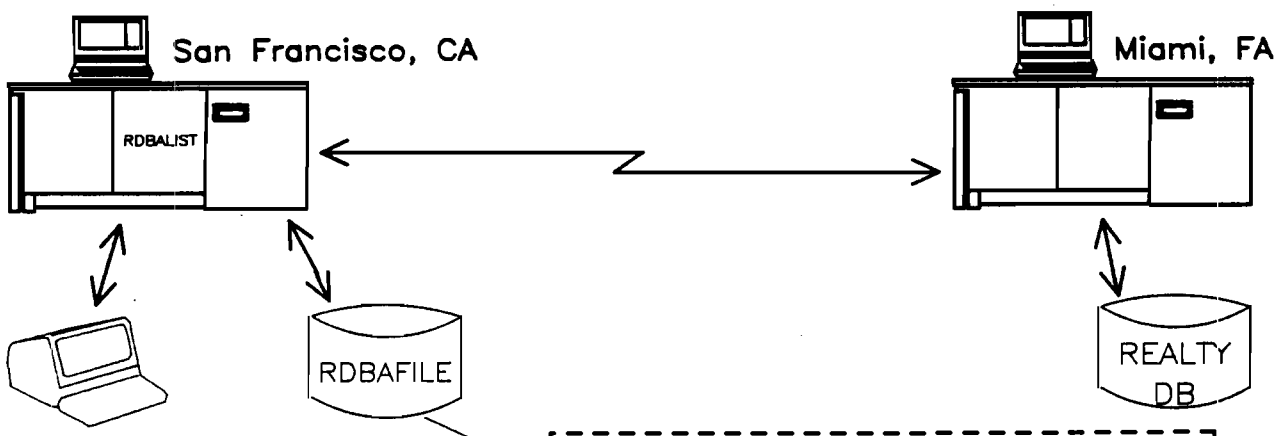
RDBA METHOD #3
PROGRAMMATIC USE OF `COMMAND'



notes:

references: MPE Intrinsics Reference Manual

RDBA METHOD #4
RDBA FILE



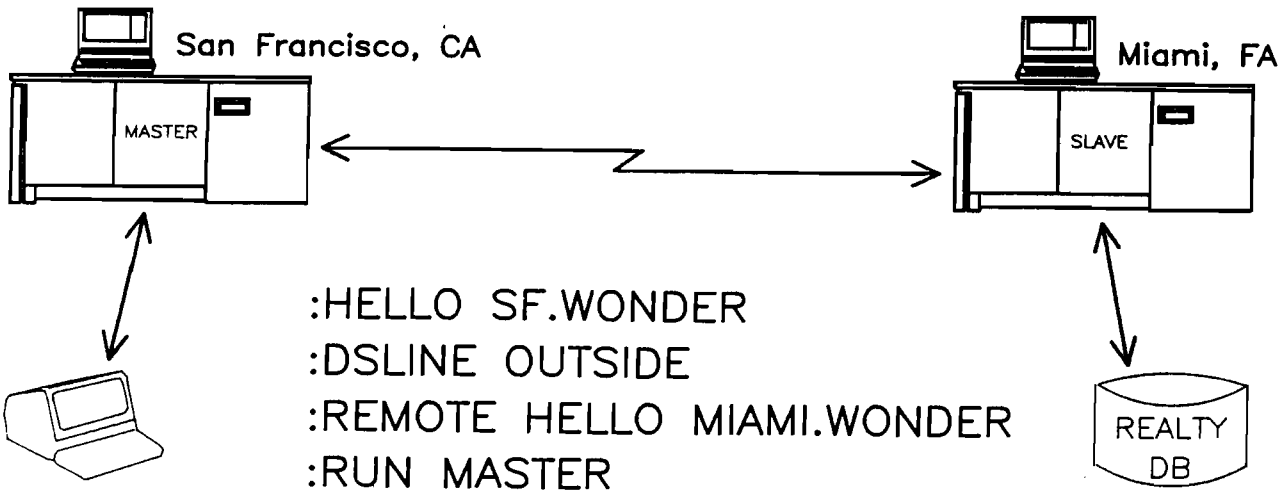
:HELLO SF.WONDER
:RUN RDBALIST
(RDBALIST DBOPEN's
RDBAFILE)

notes:

- "RDBAFILE" must be ACTIVATE'd using DBUTIL

references:

RDBA METHOD #5
PTOP



:HELLO SF.WONDER
:DSLINE OUTSIDE
:REMOTE HELLO MIAMI.WONDER
:RUN MASTER

* Creates and controls SLAVE
* SLAVE does data base access on remote HP3000

notes:

- PTOP = "Program to Program"

references:

CONSIDERATIONS FOR RDBA

* Since accessing a remote data base naturally increases response time and system overhead, use it wisely.

* Follow the 80/20 rule:

  Do 80% of the work (data base accesses) locally.

  Do no more than 20% on the remote HP3000!

notes:

references:

IMAGE

AND

RAPID/3000

notes:

references:

RAPID/3000

* DICTIONARY/3000: Data definition and directory
* TRANSACT/3000: High-level programming tool
* REPORT/3000: Programmer's report writer
* INFORM/3000: End-user's report writer

notes:

references:

DICTIONARY/3000

* Data definitions and directory for IMAGE data bases, KSAM and MPE files, VPLUS forms files, programs, reports

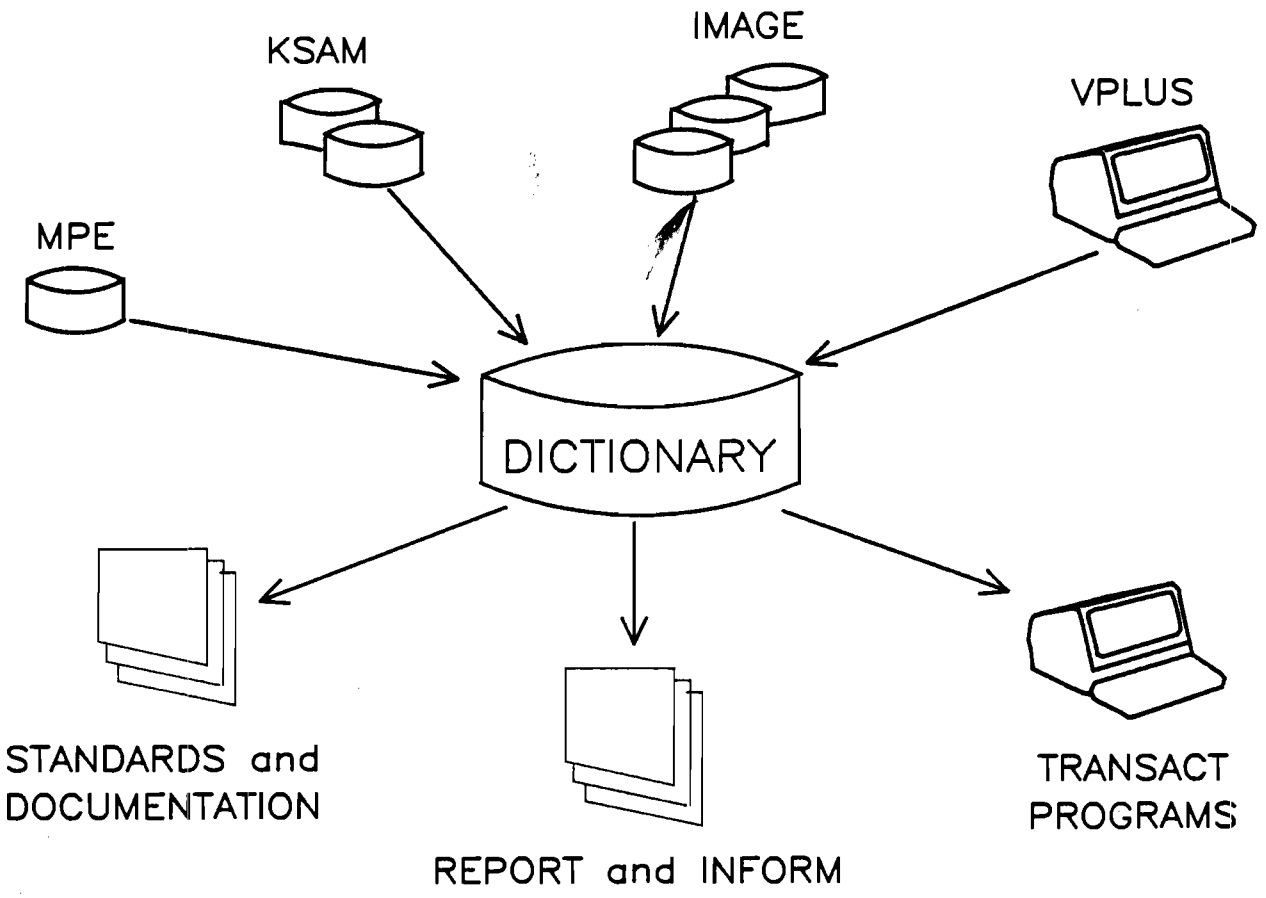
* Uses IMAGE data base to store "data about your data"

* Data definition and access information for TRANSACT, REPORT, and INFORM

notes:

references: DICTIONARY/3000 Reference Manual

DICTIONARY/3000



notes:

references:

DICTIONARY/3000

* Special utilities for use with IMAGE:
  - Create new data bases
  - Backload existing data bases into a dictionary
  - Unload and load to disc or tape, by set, with more restructuring capability
  - Audit chain lengths, synonyms

notes:

references:

DICTIONARY/3000 HP32244A.00.01 – (C) Hewlett–Packard Co. 1981:

ELEMENT:	TYPE:	SIZE:	DEC:	LENGTH:	COUNT:	RESPONSIBILITY:
CITY–ABBR	X	4	0	4	1	MGMT

LONG NAME: CITY ABBREVIATION
HEADING TEXT: CITY
ENTRY TEXT: PLEASE ENTER CITY ABBREVIATION
EDIT MASK:
MEASUREMENT UNITS:
DATE CREATED: 82/04/22 BY MGR
DATE CHANGED: 82/04/22 BY MGR

DESCRIPTION:
FOUR CHARACTER CITY ABBREVIATION

FILE:	TYPE: ELEMENT(ALIAS):
CITY–MASTER	MAST CITY–ABBR
RESIDENTIAL	DETL CITY–ABBR
	CHAIN MASTER SET: !CITY–MASTER
COMMERCIAL	DETL CITY–ABBR
	CHAIN MASTER SET: CITY–MASTER

1 RECORD FOUND

notes:

references:

TRANSACT/3000

* High level COBOL—like coding constructs
* Can be used with a dictionary to eliminate data definitions
* Excellent for speedy program development

notes:

references: TRANSACT/3000 Reference Manual

DELETE CITY-MASTER;

```
300-DELETE-MASTER.
     MOVE "CITY-MASTER;" TO DSET-NAME.
     CALL "DBGET" USING BASE DSET-NAME MODE7
          STATUS-AREA ITEM-LIST BUFFER ARG-VALUE.
     IF C-W NOT = 0
          GO TO 900-ERROR.
     CALL "DBDELETE" USING BASE DSET-NAME MODE1
          STATUS-AREA.
     IF C-W NOT = 0
          GO TO 900-ERROR.
     STOP RUN.
399-EXIT.
900-ERROR.
     CALL "DBEXPLAIN" USING STATUS-AREA.
     STOP RUN.
```

TRANSACT/3000 | COBOL II

notes:

references:

REPORT/3000

* Complex access and formatting
* Simple English-like language
* Reports from multiple sets, bases, MPE and KSAM files
* Allows "relational view" of data

notes:

references: Using REPORT/3000

REPORT/3000

```
REPORT HOUSE;
SELECT LISTING-NR;
OPTION NOHEAD;
REPORT TITLE "RESIDENTIAL LISTING",COL=4;
DETAIL "LISTING NUMBER:",LINE=3:LISTING-NR,SPACE=1:
       "LIST PRICE:",LINE=1,COL=1:LIST-PRICE,SPACE=1:
       "# BEDROOMS:",LINE=1:NUMBER-BEDS,SPACE=1:
       "# BATHS:",LINE=1:NUMBER-BATHS,SPACE=1:
       "DINING ROOM?",LINE=1:FORMAL-DINING-RM,SPACE=1:
       "SOLD?",LINE=1:SOLD-FLAG,SPACE=1:
       "SQUARE FEET:",LINE=1:SQUARE-FEET,SPACE=1:
       "CITY:",LINE=1:CITY-ABBR,SPACE=1;
END;
```

SOURCE

```
                RESIDENTIAL LISTING

LISTING NUMBER:      21
LIST PRICE:     $162K
# BEDROOMS:  3
# BATHS: 2.00
DINING ROOM? N
SOLD? N
SQUARE FEET:     1245
CITY: MH


LISTING NUMBER:      22
LIST PRICE:     $229K
#BEDROOMS:  4
# BATHS: 3.00
DINING ROOM? Y
SOLD/ N
SQUARE FEET:     1439
CITY: MH
```

REPORT

notes:

references:

INFORM/3000

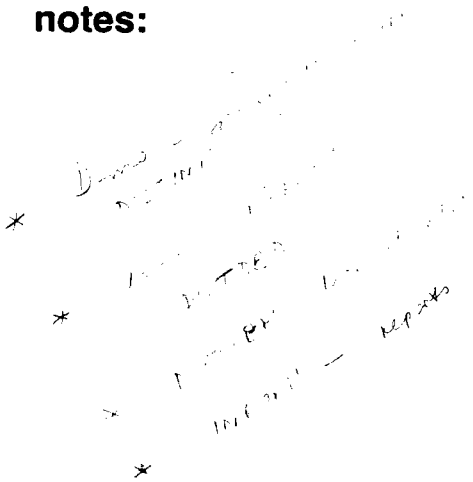
* End—users can build their own reports quickly and easily

* DICTIONARY/3000 provides customized menus for the user

* Reports from multiple sets, bases, MPE and KSAM files

* Allows "relational view" of data via DICTIONARY/3000

notes:



references: Using INFORM/3000

DATA NAMES IN DATA SET RESIDENTIAL

1: LISTING–NR	5: NUMBER–BATHS	9: ZIP–CODE
2: CITY–ABBR	6: CURRENT–OWNER	10: FORMAL–DINING–RM
3: LIST–PRICE	7: OWNERS–PHONENR	11: SOLD–FLAG
4: NUMBER–BEDS	8: STREET–ADDRESS	12: SQUARE–FEET

TYPE NUMBER(S) FOR DATA NAME(S):

TO INCLUDE IN REPORT> 1,3,8,2
TO SORT BY> 3,1
TO SUBDIVIDE BY>
TO GRAND TOTAL>
FOR SELECTION CRITERIA> 3

REPORT TITLE> RESIDENTIAL LISTING REPORT

MENU

REPORT

MON, JUL 26, 1982, 4:18 PM

RESIDENTIAL LISTING REPORT

LISTING NO.	PRICE	ADDRESS	CITY
29	$161K	5433 DOVER DR	SC
33	$162K	5600 PARTER RD	SJ
21	$162K	534 BLACKFORD RD	MH
5	$167K	8876 HARTE AVE	CAMB
32	$168K	5566 ULTIMA DR	SJ
⋮	⋮	⋮	⋮

notes:

references:

```
$TITLE "WONDER REALTY DATA BASE"

BEGIN DATA BASE REALTY;

PASSWORDS:

   10       RECEPT;                        << RECEPTIONIST              >>
   20       SALESREP;                      << SALES PERSON              >>
   30       MANAGER;                       << BOSS PERSON               >>

ITEMS:

   CITY-ABBR,           X4    (10,20/30); << CITY ABBREVIATION         >>
   CITY-NAME,           X20   (10,20/30); << CITY NAME                 >>
   LIST-PRICE,          I     (10,20/30); << LIST PRICE TO NEAREST $1K>>
   CURRENT-OWNER,       X20   (20/30);    << CURRENT OWNER             >>
   SOLD-FLAG,           X2    (10/20,30); << "Y" OR BLANK              >>
   SQUARE-FEET,         X8    (10,20/30); << SQUARE FEET               >>

$TITLE "MASTER DATA SETS"

SETS:

$CONTROL BLOCKMAX=128

   NAME:        LIST-PRICE-MSTR, AUTOMATIC (10,20/30);
   ENTRY:       LIST-PRICE(1);
   CAPACITY:    307;

   NAME:        CITY-MASTER, MANUAL (10,20/30);
   ENTRY:       CITY-ABBR (1),
                CITY-NAME;
   CAPACITY:    101;

$TITLE "DETAIL DATA SETS"

$CONTROL BLOCKMAX=640

   NAME:        RESIDENTIAL, DETAIL (10,20/30);
   ENTRY:       CITY-ABBR    (!CITY-MASTER),
                LIST-PRICE   (LIST-PRICE-MSTR (SQUARE-FEET)),
                CURRENT-OWNER,
                SOLD-FLAG,
                SQUARE-FEET;
   CAPACITY:    300;

END.
```

```
PAGE 1     HEWLETT-PACKARD 32215B.03.00 IMAGE/3000: DBSCHEMA  MON, JUL 19, 1982

$TITLE "WONDER REALTY DATA BASE"

BEGIN DATA BASE REALTY;

$CONTROL NOLIST

ITEMS:

   CITY-ABBR,           X4   (10,20/30); << CITY ABBREVIATION          >>
   CITY-NAME,           X20  (10,20/30); << CITY NAME                  >>
   LIST-PRICE,          I    (10,20/30); << LIST PRICE TO NEAREST $1K>>
   CURRENT-OWNER,       X20  (20/30);    << CURRENT OWNER              >>
   SOLD-FLAG,           X2   (10/20,30); << "Y" OR BLANK               >>
   SQUARE-FEET,         X8   (10,20/30); << SQUARE FEET                >>

$TITLE "MASTER DATA SETS"

SETS:

$CONTROL BLOCKMAX=128

   NAME:        LIST-PRICE-MSTR, AUTOMATIC (10,20/30);
   ENTRY:       LIST-PRICE(1);
   CAPACITY:    307;

   NAME:        CITY-MASTER, MANUAL (10,20/30);
   ENTRY:       CITY-ABBR (1),
                CITY-NAME;
   CAPACITY:    101;

$TITLE "DETAIL DATA SETS"

$CONTROL BLOCKMAX=640

   NAME:        RESIDENTIAL, DETAIL (10,20/30);
   ENTRY:       CITY-ABBR    (!CITY-MASTER),
                LIST-PRICE   (LIST-PRICE-MSTR (SQUARE-FEET)),
                CURRENT-OWNER,
                SOLD-FLAG,
                SQUARE-FEET;
   CAPACITY:    300;

END.

 DATA SET       TYPE  FLD  PT  ENTR  MED  CAPACITY  BLK  BLK   DISC
   NAME               CNT  CT  LGTH  REC            FAC  LGTH  SPACE

LIST-PRICE-MSTR    A   1    1   1     11    307      11   122   30
CITY-MASTER        M   2    1   12    22    101      5    111   23
RESIDENTIAL        D   5    2   18    26    312      24   626   70
                          TOTAL DISC SECTORS INCLUDING ROOT: 132
```

```
$TITLE "WONDER REALTY DATA BASE"

BEGIN DATA BASE REALTY;

PASSWORDS:

   10       RECEPT;                    << RECEPTIONIST              >>
   20       SALESREP;                  << SALES PERSON              >>
   30       MANAGER;                   << BOSS PERSON               >>

ITEMS:

   LISTING-NR,          I    (10,20/30); << UNIQUE, SEQUENTIAL NUMBER>>
   NUMBER-BEDS,         X2   (10,20/30); << NUMBER OF BEDROOMS       >>
   CITY-ABBR,           X4   (10,20/30); << CITY ABBREVIATION        >>
   CITY-NAME,           X20  (10,20/30); << CITY NAME                >>
   ZONING-CODE,         X4   (10,20/30); << ZONING CODE              >>
   ZONING-DESCRIP,      X20  (10,20/30); << ZONING DESCRIPTION       >>
   NUMBER-BATHS,        X4   (10,20/30); << NUMBER OF BATHS  (d.dd)  >>
   LIST-PRICE,          I    (10,20/30); << LIST PRICE TO NEAREST $1K>>
   CURRENT-OWNER,       X20  (20/30);    << CURRENT OWNER            >>
   OWNERS-PHONENR,      X10  (20/30);    << OWNERS PHONE NUMBER      >>
   STREET-ADDRESS,      X30  (20/30);    << STREET ADDRESS           >>
   ZIP-CODE,            X6   (10,20/30); << ZIP CODE (5 DIGIT)       >>
   FORMAL-DINING-RM,    X2   (10,20/30); << "Y" OR BLANK             >>
   SOLD-FLAG,           X2   (10/20,30); << "Y" OR BLANK             >>
   SQUARE-FEET,         X8   (10,20/30); << SQUARE FEET              >>
   EXISTING-STRUCT,     X2   (10,20/30); << "Y" OR BLANK             >>
   OCCUPIED,            X2   (10,20/30); << "Y" OR BLANK             >>

$TITLE "MASTER DATA SETS"

SETS:

$CONTROL BLOCKMAX=128

   NAME:        LISTNR-MASTER, AUTOMATIC (10,20/30);
   ENTRY:       LISTING-NR(2);
   CAPACITY:    503;

   NAME:        LIST-PRICE-MSTR, AUTOMATIC (10,20/30);
   ENTRY:       LIST-PRICE(2);
   CAPACITY:    307;

   NAME:        BATH-MASTER, AUTOMATIC (10,20/30);
   ENTRY:       NUMBER-BATHS(1);
   CAPACITY:    31;

   NAME:        BEDS-MASTER, AUTOMATIC (10,20/30);
   ENTRY:       NUMBER-BEDS (1);
   CAPACITY:    11;
```

```
   NAME:          ZONING-MASTER, MANUAL (10,20/30);
   ENTRY:         ZONING-CODE (1),
                  ZONING-DESCRIP;
   CAPACITY:      31;

   NAME:          CITY-MASTER, MANUAL (10,20/30);
   ENTRY:         CITY-ABBR (2),
                  CITY-NAME;
   CAPACITY:      101;

$TITLE "DETAIL DATA SETS"

$CONTROL BLOCKMAX=640

   NAME:          RESIDENTIAL, DETAIL (10,20/30);
   ENTRY:         LISTING-NR    (LISTNR-MASTER),
                  CITY-ABBR     (!CITY-MASTER),
                  LIST-PRICE    (LIST-PRICE-MSTR (SQUARE-FEET)),
                  NUMBER-BEDS   (BEDS-MASTER),
                  NUMBER-BATHS  (BATH-MASTER),
                  CURRENT-OWNER,
                  OWNERS-PHONENR,
                  STREET-ADDRESS,
                  ZIP-CODE,
                  FORMAL-DINING-RM,
                  SOLD-FLAG,
                  SQUARE-FEET;
   CAPACITY:      300;

$CONTROL BLOCKMAX=768

   NAME:          COMMERCIAL, DETAIL  (10,20/30);
   ENTRY:         LISTING-NR     (LISTNR-MASTER),
                  ZONING-CODE    (!ZONING-MASTER),
                  LIST-PRICE     (LIST-PRICE-MSTR (SQUARE-FEET)),
                  CITY-ABBR      (CITY-MASTER),
                  STREET-ADDRESS,
                  ZIP-CODE,
                  EXISTING-STRUCT,
                  OCCUPIED,
                  SOLD-FLAG,
                  SQUARE-FEET;
   CAPACITY:      300;

END.
```

APPENDIX B

READING/REFERENCE MATERIAL ON DATA BASE MANAGEMENT

-- HP IMAGE/3000 Reference Manual

-- HP QUERY/3000 Reference Manual

-- "An Introduction to Data Base Systems -- 2nd Edition", C. J. Date, Addison-Wesley, Menlo Park, CA.

-- "Principles of Data Base Management", James Martin, Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632

-- "Computer Data Base Organization, 2nd Edition", James Martin, Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632

-- "Practical Data Base Management", Edited by Auerbach Publishers, Reston Publishing Co., Inc., Reston, Virginia

-- "Data Base: Structured Techniques for Desin, Performance, and Management", S. Atre, John Wiley & Sons

APPENDIX C

PARAMETER DATA TYPES

Examples for Specific Programming Languages

* Array

COBOL
```
01 DSET-NAME   PIC X(16)  VALUE "CITY-MASTER;"
```

BASIC
```
DIM  D$[16]
D$ = "CITY-MASTER;"
```

FORTRAN
```
DIMENSION DSETA(8)
CHARACTER*16 DSETC
EQUIVALENCE (DSETA(1),DSETC)
DSETC = "CITY-MASTER;"
```

PASCAL
```
dset_type = PACKED ARRAY [1..16] OF CHAR
dset_name : dset_type
dset_name := 'CITY-MASTER;'
```

SPL
```
BYTE ARRAY DSETNAME (0:16) := "CITY-MASTER;"
```

* Integer

COBOL
```
01 MODE   PIC 9999   COMP VALUE 1.
```

BASIC
```
INTEGER M
M = 1
```

FORTRAN
```
MODE = 1
```

PASCAL
```
single_integer = -32768..32767
mode : single_integer
mode := 1
```

SPL
```
INTEGER MODE := 1
```

* Double Integ

COBOL
```
01 REC-NUM   PIC S9(9)   COMP.
```

BASIC
```
INTEGER R[2]
```

FORTRAN
```
INTEGER*4 RECNUM
```

PASCAL
```
rec_num : integer
```

SPL
```
DOUBLE RECNUM
```

* Integer Array

COBOL
```
01  STATUS.
    05 COND-WORD   PIC  S9(4)  COMP.
    05 STAT1       PIC  S9(4)  COMP.
    05 STAT2-3     PIC  S9(9)  COMP.
    05 STAT4-5     PIC  S9(9)  COMP.
    05 STAT6-7     PIC  S9(9)  COMP.
    05 STAT8-9     PIC  S9(9)  COMP.
```

BASIC
```
INTEGER S[10]
```

FORTRAN
```
DIMENSION STATUS(10)
```

PASCAL
```
single_integer = -32768..32767
status = ARRAY[1..10] OF single_integer
```

SPL
```
INTEGER ARRAY STATUS(0:9)
```

APPENDIX D

RPG & LOCKING

A variety of user-controlled file locking capabilities have been implemented. This enhancement allows you to perform conditional or unconditional locking and unlocking when you want to on Image, KSAM, or MPE files. For an Image file, you may perform locks at the data base, data set, or record level. For a KSAM and MPE file, you may only lock at the file level.

An unconditional lock is a lock request on a data object such that if the data object is already locked by another process, the lock request will wait in a queue of all lock requests for the data object until it is its term to lock. All the while the process which requested the unconditional lock is suspended until the lock request is granted. The time the process is suspended may seriously degrade performance in an interactive processing environment. To overcome this, conditional locks should be performed.

A conditional lock is a lock request on a data object such that if the object is already locked by another process, the lock will fail and a resulting Indicator will turn on to inform you of the situation. The process is not suspended, so it may now proceed to perhaps process a different data object, returning later when this data object is available.

To perform all locking and unlocking yourself, you use the new calculation operations LOCK and UNLCK. In the discussion below, every use of LOCK also refers to UNLCK.

LOCKING AN IMAGE DATA BASE:

To lock an Image data base, you specify "LOCK" in the calculation operation field, a file name in Factor 2, a data base name in the Result Field, and Result Indicators.

The file name specified in Factor 2 must be a file name given in your File Specifications which describes the Image data base to be locked with K extensions.

The data base name specified in the Result Field must be the data base name given in the KIMAGE specifications for the file in Factor 2. The data base name here in the Result Field is the data base you wish to lock. It must be a literal string, not a variable. THE NUMBER OF CHARACTERS IN THE DATA BASE NAME MUST BE SPECIFIED IN THE RESULT LENGTH FIELD (col. 49-51).

The KIMAGE specifications for the Image data base must also specify "L" (enable locking) as the Image open mode in column 66.

Result Indicators must be specified for this operation. The High Indicator is optional, but one of either the Low or Equal Indicators is required. The presence of the High Indicator specifies conditional locking, whereas its absence specifies unconditional. The Result Indicators return the following status information:

Result Indicator ON Status Information

High	>	lock failed - already locked by another process (conditional)
Low	<	lock failed - not opened with locking enabled or need MR CAP
Equal	=	successful lock

LOCKING AN IMAGE DATA SET

To lock an Image data set, you specify "LOCK" in the calculation operation field, a filename in Factor 2, and Result Indicators. The Result Field must be blank.

The file name specified in Factor 2 is the name of the data set to be locked as described in the File Description Specifications. Furthermore, the KIMAGE specifications for the data base which contains the data set must specify "L" (enable locking) as the Image open mode in column 66.

Result Indicators must be specified, and their individual purposes are the same as explained above in Locking an Image data base.

LOCKING AN IMAGE RECORD

To lock an Image record, you specify "LOCK" in the calculation operation field, a search key in Factor 1, a data set file name in Factor 2, and Result Indicators. The Result Field must be blank.

The key specified in Factor 1 is used as the search key for the record(s) which contain that key. All records with items that equal the search key will be locked.

The file name in Factor 2 is the name of the data set you wish to search for records to lock. This file must be described in the File Specifications with KIMAGE and KITEM extensions. The KIMAGE specifications must specify "L" as the Image open mode in column 66 to enable locking. KITEM specifies the IMAGE item field to be used against the search key.

Result Indicators are required and are the same as explained above in the first section on Locking an Image Data Base.

LOCKING A KSAM OR MPE FILE

To lock a KSAM file or MPE file, you specify "LOCK" in the calculation operation field, the file name in Factor 2, and Result Indicators.

The file name in Factor 2 must be a name given in the File Description Specifications. If the file is described there as a KSAM file, then this is a KSAM lock operation, else it is an MPE file lock.

The File Specifications for the file to be locked must specify a NOLOCK extension to enable locking. Furthermore, Result Indicators are required and are the same as explained above in the first section on Image data base locking.

PROGRAMMING CAUTIONS

1. Multiple RIN Capability

   If you wish to request more than one outstanding lock, you must have MR special capability. See the Image Reference Manual for a discussion on MR CAP.

2. Dsname'd Files

   If you request RPG to perform all locking for you on a cycle by cycle basis, you cannot do your own locking in Calculation Spec.

NEW ERROR MESSAGES

ERROR 681T - LOW OR EQUAL RESULT INDICATOR MUST BE SPECIFIED FOR THIS OPERATION

ERROR 682T - IMAGE FILE LOCKING MODE IS NOT L TO ENABLE LOCKING ONLY

ERROR 683T - FILE NOT DESCRIBED WITH NOLOCK TO ENABLE LOCKING

ERROR 684T - DB NAME IN RESULT FIELD MUST BE SAME NAME SPECIFIED IN IMAGE SPEC FOR FILE IN FACTOR 2

SUMMARY TABLE OF LOCK/UNLCK

Type of Lock	Factor 1	Oper	Factor 2	Result Fld
Image Record	key	LOCK	filename	blank
Image DataSet	blank	LOCK	filename	blank
Image DataBase	blank	LOCK	filename	data base
KSAM file	blank	LOCK	filename	blank
MPE file	blank	LOCK	filename	blank

Result Indicator ON Status Information

High	>	lock failed - already locked by another process (conditional)
Low	<	lock failed - not opened with locking enabled or need MR CAP
Equal	=	successful lock

EXAMPLES

FILE DESCRIPTION SPECIFICATIONS

```
FINFILE   IP  F     80           DISC
FKSDATA1  IC  F     72R14AI   51 DISC
F                                          KIMAGE EMP1  L5
F                                          KITEM  CITY
```

CALCULATION SPECIFICATIONS

```
**  IMAGE DATA BASE LEVEL LOCKING
**
C                       LOCK KSDATA1    EMP1    4  737475
C  75     CITKEY        CHAINKSDATA1                8058
C  75                   UNLCKKSDATA1    EMP1    4     76
```

SOURCE:

HP 3000 Communicator, MPE III 2028, Issue number 25, 7/80
Page 29, #5. "Enhancements to RPG/3000"

APPENDIX E

IMAGE UTILITY ERROR MESSAGES

INTRODUCTION:

An analysis of the cryptic error messages displayed when IMAGE'S DBLOAD, DBUNLOAD, DBSTORE and DBRESTOR fail.

When the IMAGE utilities that back-up and restore data base files cannot complete their job, they sometimes issue a cryptic error message. This document is a compilation of the possible error codes with their meanings, and any specific actions you can take to solve the problem. For many cases, IMAGE is simply reporting a system problem, and all that can be supplied here is a guide to be used in interpretting IMAGE'S message--the solution being dependent on the problem being reported.

Error Message Format DATA BASE UTILITY ERROR P1/P2/P3/P4/P5

P1-P3 For DBUNLOAD/DBLOAD these three numbers are usually zero. For DBSTORE/DBRESTOR they are used to further define error.

P4 This is the key to de-coding the information in the 5 numbers which are displayed. The various values for P4 are listed below with their meaning.

P5 This is the octal address of the IMAGE code that requested the error message.

DBUNLOAD

P4	INTERPRETATION
4xx	An IMAGE intrinsic failed and returned an un-identifiable error code. See the IMAGE manual to relate the number to a specific intrinsic.

DBLOAD

P4	INTERPRETATION
-63	The number of data-sets on tape does not equal the number of data sets expected. In order to add or delete data-sets from a data base, DBLOAD must be run with PARM=1234. This PARM inhibits the initial error checking that normally detects a change in the number of data-sets. This PARM also inhibits the comparison of the log-on GROUP.ACCOUNT with the tape-file's GROUP.ACCOUNT, thus allowing a data base to be moved across group and account boundries.

-64 The data on the tape is out of sequence. This error can occur if

1) the tape (or tape drive) is bad, which prevents accurate reading of the reel.

2) in a multi-reel environment, data is written before the load-point on a reel other than the first. This may be the case if the error occurred immediately after the reel was mounted. Use FCOPY to display the first several records of the reel. If the first eight characters on the tape are anything but TAPEHEAD, data has been written before the load point.

To recover, mount the tape with as little lead as possible on the take-up reel. Then put another load point marker just before the read/write heads. Press LOAD and ON-LINE and use FCOPY again. TAPEHEAD should now be the first record. ERASE the data base and re-run DBLOAD STARTING WITH THE FIRST REEL.

-74 The tape record just read is unrecognizable. (The tape cannot be read accurately.)

4xx Same as DBUNLOAD.

DBSTORE/DBRESTOR

P4	INTERPRETATION
0,100	IMAGE failure due to File System failure. The intrinsics that perform stores/restores cannot be completed. P4=0 is returned from DBRESTOR, AND P4=100 is returned from DBSTORE. In both cases P2 is the Command Interpreter error number, and P1 is the subsystem error number. The numeric index in the Error Messages & Recovery Manual is useful in interpreting these.
97,98	IMAGE failure due to ZSIZE failure. P2 and P3 show the current environment. P1=0.
-3	IMAGE failure due to FREADIR failure. P1=-3; P2=0; P3=the FCHECK error code.
-4	IMAGE failure due to FREADLABEL failure. P1=-4; P2=0; P3=the FCHECK error code.
1-99	IMAGE failure due to File System Error. P1-P3 can be interpreted as above in error 0,100; however P4 will be the number of the data-set that could not be processed successfully.

APPENDIX F

HOW TO DETERMINE NUMBER OF MPE LOG RECORDS REQUIRED PER IMAGE CALL

DBOPEN — Always 2 records (OPENLOG and WRITELOG)

DBCLOSE — 2 records (WRITELOG and CLOSELOG) and up to 31 null records (may log 1 extra record if process is aborting with unfinished transaction)

DBPUT
DBUPDATE
DBDELETE
DBMEMO
DBBEGIN
DBEND

1 or more records, determined by the formula:

$$RECS = \lceil ILEN/(128-10) \rceil$$

where ILEN is the length of the IMAGE log record

HOW TO DETERMINE IMAGE LOG RECORD LENGTHS

DBPUT

length = 14 + itemlist + bufferlength

DBUPDATE

length = 15 + keylength + itemlist + olddata + newdata

DBDELETE

length = 13 + recordlength

DBBEGIN, DBEND, DBMEMO

length = 7 + user buffer length

NOTE:

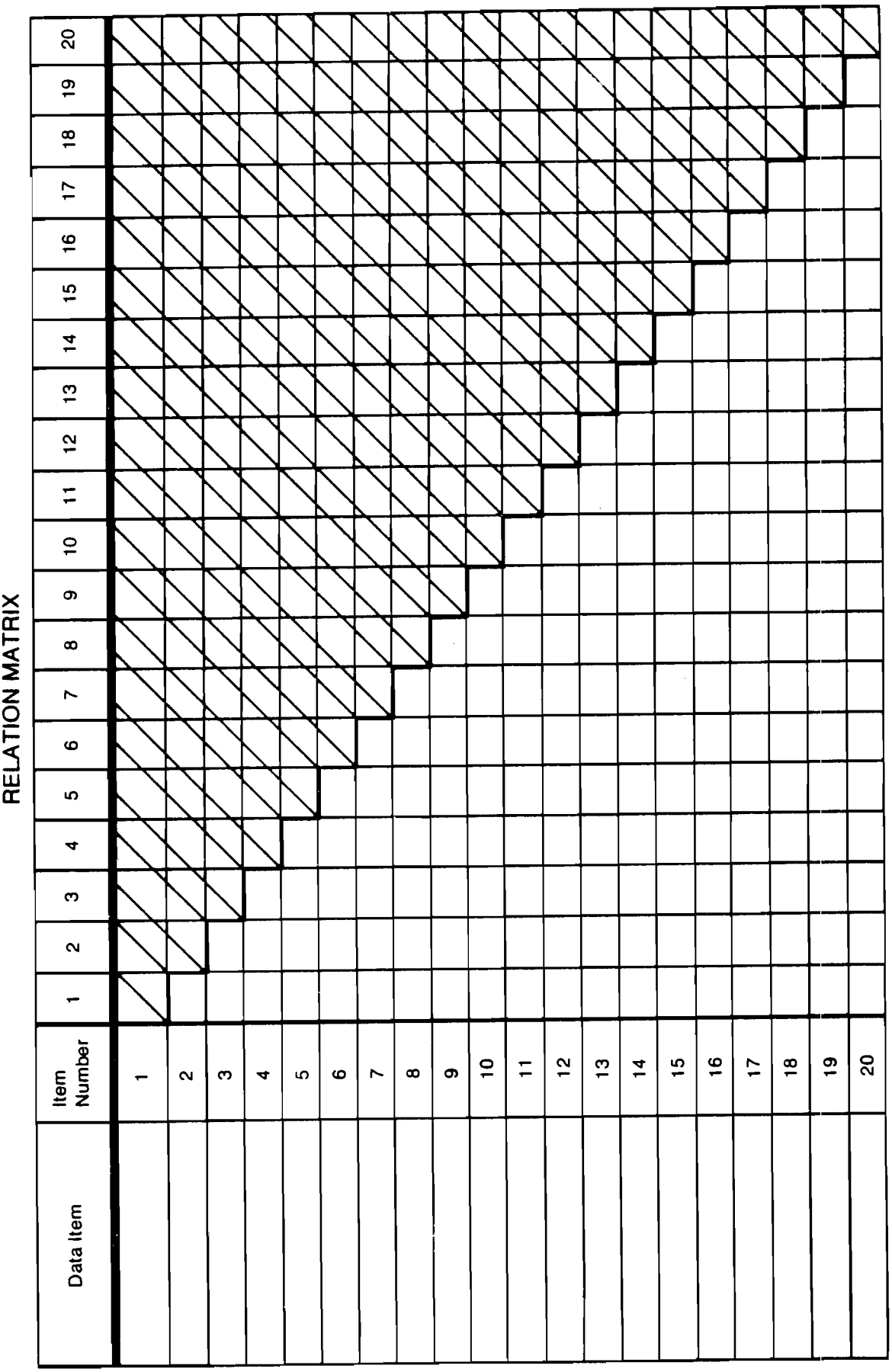
* Keylength is zero if detail data set
* Itemlists are always numeric
* DBUPDATE only logs modified data
* All calculations are in WORDS

APPENDIX G

FUNCTION MATRIX

Function Number:		1	2	3	4	5	6	7	8
Function Description:									
Data Item	Item Number	DATA ITEM FUNCTIONS							
	1								
	2								
	3								
	4								
	5								
	6								
	7								
	8								
	9								
	10								
	11								
	12								
	13								
	14								
	15								
	16								
	17								
	18								
	19								
	20								

RELATION MATRIX



35035-90001
Printed in U.S.A. 8/82

HEWLETT PACKARD