DIRECTORIO DE PROFESORES DEL CURSO:

TELLER DE DISEÑO DE BASE DE DATOS

1. ING. DANIEL RIOS ZERTUCHE
   DIRECTOR DE INFORMATICA
   SUBSECRETARIA DE PLANEACION DEL DESARROLLO
   SECRETARIA DE PROGRAMACION Y PRESUPUESTO
   IZAZAGA NO. 38-11° PISO
   MEXICO, D.F.
   521 98 98

'edcs.

TALLER DE DISEÑO DE BASE DE DATOS 1 9 8 5

Fecha	TEMA	HORARIO	PROFESOR
Del 6 al 21 de Septiembre.	**INTRODUCCION** Descripción del medio ambiente en el que se va a trabajar.Utilización del equipo **INTRODUCCION AL DBMS** Descripción del manejador de base de datos relacional a utilizar, sus utilerías y lenguajes de consulta. **INTRODUCCION A LOS LENGUAJES DE CUARTA GENERACION** Generadores de aplicaciones Lenguajes de Muy Alto Nivel Lenguajes no procedurales Lenguajes de Consulta. **PROTOTIPOS** La metodología de los prototipos y su interrelación con las metodologías en Base al Ciclo de Vida del Software. **EL CENTRO DE INFORMACION** El concepto de Centro de Información sus ventajas y desventajas y su interrelación con el Centro de Proceso de Datos Tradicional. **APLICACIONES** Negocios Industrias Educación	Viernes de 17 a 21 h Sábados de 9 a 14 h	ING.DANIEL RIOS ZERTUCI

E V A L U A C I O N D E L P E R S O N A L

D O C E N T E

CURSO: TALLER DE DISEÑO DE BASE DE DATOS

FECHA: Del 6 al 21 de septiembre de 1985.

C O N F E R E N C I S T A	DOMINIO DEL TEMA	EFICIENCIA EN EL USO DE AYUDAS AUDIOVISUALES	MANTENIMIENTO DEL INTERES. (COMUNICACION CON LOS ASISTENTES, AMENIDAD, FACILIDAD DE EXPRESION)	PUNTUALIDAD	
ING.DANIEL RIOS ZERTUCHE					

ESCALA DE EVALUACION: 1 A 10

E V A L U A C I O N D E L A E N S E Ñ A N Z A

CURSO:

FECHA:

TEMA	ORGANIZACION Y DESARROLLO DEL TEMA	GRADO DE PROFUNDIDAD LOGRADO EN EL TEMA	GRADO DE ACTUALIZACION LOGRADO EN EL TEMA	UTILIDAD PRACTICA DEL TEMA	
INTRODUCCION					
INTRODUCCION AL DBMS					
INTRODUCCION A LOS LENGUAJES DE...					
PROTOTIPOS					
EL CENTRO DE INFORMACION					
APLICACIONES					

ESCALA DE EVALUACION: 1 A 10

EVALUACION DEL CURSO

	CONCEPTO	
1.	APLICACION INMEDIATA DE LOS CONCEPTOS EXPUESTOS	
2.	CLARIDAD CON QUE SE EXPUSIERON LOS TEMAS	
3.	GRADO DE ACTUALIZACION LOGRADO EN EL CURSO	
4.	CUMPLIMIENTO DE LOS OBJETIVOS DEL CURSO	
5.	CONTINUIDAD EN LOS TEMAS DEL CURSO	
6.	CALIDAD DE LAS NOTAS DEL CURSO	
7.	GRADO DE MOTIVACION LOGRADO EN EL CURSO	

ESCALA DE EVALUACION: 1 A 10

1.- ¿Qué le pareció el ambiente en la División de Educación Continua?

MUY AGRADABLE ☐ AGRADABLE ☐ DESAGRADABLE ☐

2.- Medio de comunicación por el que se enteró del curso:

PERIODICO EXCELSIOR ANUNCIO TITULADO DIVISION DE EDUCACION CONTINUA ☐

PERIODICO NOVEDADES ANUNCIO TITULADO DIVISION DE EDUCACION CONTINUA ☐

FOLLETO DEL CURSO ☐

CARTEL MENSUAL ☐

RADIO UNIVERSIDAD ☐

COMUNICACION CARTA, TELEFONO, VERBAL, ETC. ☐

REVISTAS TECNICAS ☐

FOLLETO ANUAL ☐

CARTELERA UNAM "LOS UNIVERSITARIOS HOY" ☐

GACETA UNAM ☐

3.- Medio de transporte utilizado para venir al Palacio de Minería:

AUTOMOVIL PARTICULAR ☐

METRO ☐

OTRO MEDIO ☐

4.- ¿Qué cambios haría en el programa para tratar de perfeccionar el curso?

_____

_____

_____

5.- ¿Recomendaría el curso a otras personas?

☐ sí ☐ no

6.- ¿Qué cursos le gustaría que ofreciera la División de Educación Continua?

_____

_____

_____

7.- La coordinación académica fué:

EXCELENTE ☐ BUENA ☐ REGULAR ☐ MALA ☐

8.- Si está interesado en tomar algún curso INTENSIVO ¿Cuál es el horario más conveniente para usted?

LUNES A VIERNES DE 9 a 13 H. Y DE 14 A 18 H. (CON COMIDAD) ☐

LUNES A VIERNES DE 17 a 21 H. ☐

LUNES A MIERCOLES Y VIERNES DE 18 A 21 H. ☐

MARTES Y JUEVES DE 18 A 21 H. ☐

VIERNES DE 17 A 21 H. SABADOS DE 9 A 14 H. ☐

VIERNES DE 17 A 21 H. SABADOS DE 9 A 13 H. DE 14 A 18 H. ☐

OTRO ☐

9.- ¿Qué servicios adicionales desearía que tuviese la División de Educación Continua, para los asistentes?

_____

_____

_____

10.- Otras sugerencias:

_____

_____

_____

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

TALLER DE DISEÑO DE BASES DE DATOS

ING. DANIEL RIOS ZERTUCHE

SEPTIEMBRE, 1985.

RESUMEN EN ESPANOL DE LA BASE DE DATOS INFORMIX

C O N T E N I D O

1. INTRODUCCION DE INFORMIX — hoja

I.	Que es un Sistema Manejador de Base de Datos?	---	4
II.	Que es un Sistema de Base de Datos Relacional?	---	4

2. DBBUILD

I.	Introduccion	---	7
II.	Base de Datos STORES	---	7
III.	Definicion de DBBUILD	---	9
IV.	Indices	---	11
V.	Llaves Primarias	---	11
VI.	Ejemplo de una Corrida	---	12
VII.	Cambios a la Estructura de una Base de Datos	---	12

3. ENTER2

I.	Introduccion	---	13
II.	Ejecucion de ENTER2	---	14
III.	Hechos Sobresalientes	---	15
IV.	Chequeo de Datos Interactivamente	---	15
V.	Busquedas Genericas	---	15
VI.	Los Comandos y sus Usos	---	15

4. INFORMER

I.	Introduccion	---	19
II.	Sintesis y Estructura	---	19
III.	Comandos y Definiciones	---	20
IV.	El Estatuto EXECUTE	---	23
V.	Sintaxis	---	24

5. DBSTATUS

I.	Introduccion	---	28
II.	Adicion y Eliminacion de Indices	---	29
III.	Control de Cambios de un Archivo	---	29
IV.	Eliminacion de Archivos y Base de Datos	---	29
V.	Reproduccion del Esquema de la Base de Datos	---	29
VI.	Definicion del Comando DBSTATUS	---	29
VII.	El Comando PRINT SCHEMA	---	30
VIII.	El Comando PRINT STATUS	---	31
IX.	El Comando ADD INDEX	---	32
X.	El Comando DELETE	---	32

XI.	Los Comandos ERASE-DATABASE-FILE-AUDIT TRAIL	---	33
XII.	El Comando RENAME FIELD	---	34
XIII.	El Comando START AUDIT	---	34
XIV.	El Comando STOP AUDIT	---	34
XV.	El Comando RECOVER FILE	---	35
XVI.	El Comando LOAD	---	35
XVII.	El Comando UNLOAD	---	35
XVIII.	El Comando SELECT DATABASE	---	36
XIX.	El Comando HELP	---	36
XX.	El Comando BYE	---	37

6. REPORT WRITER

I.	Introduccion	---	38
II.	Ejemplos de Reportes	---	38
III.	Clasificacion de la Salida	---	40
IV.	El Comando Format	---	41
V.	Calculos	---	42
VI.	Agrupacion de Datos en el Reporte	---	43
VII.	Uso de Datos de 2 Archivos	---	45
VIII.	Como Escribir y Ejecutar los Programas ACE	---	46
IX.	Ejecucion de un Programa ACE	---	47
X.	Impresion de la Fecha y la Hora	---	48
XI.	Reportes Usando Parametros	---	48
XII.	Uso de Variables para Subcalculos	---	49
XIII.	Control de la Impresion con IF-THEN-ELSE	---	51
XIV.	Pausa en la Presentacion de la Informacion	---	51
XV.	Impresion con el WITHOUT TRAILING BLANKS	---	51
XVI.	Salto a la Parte Superior de la Pagina	---	52
XVII.	Definicion del Lenguaje ACE	---	52
XVIII.	Numeros	---	53
XIX.	Estructura de Identificadores y Variables	---	53
XX.	El Comando Database	---	53
XXI.	El Comando Define	---	54
XXII.	El Comando Input	---	54
XXIII.	El Comando Output	---	55
XXIV.	El Comando Read	---	55
XXV.	El Comando Sort	---	56
XXVI.	El Comando Format	---	57
XXVII.	La Clausula FIRST PAGE HEADER	---	57
XXVIII.	La Clausula PAGE HEADER	---	57
XXIX.	La Clausula ON EVERY RECORD	---	57
XXX.	Las Clausulas BEFORE GROUP OF y AFTER GROUP OF	---	57
XXXI.	La Clausula ON LAST RECORD	---	57
XXXII.	La Clausula PAGE TRAILER	---	57
XXXIII.	Impresion de Archivos	---	58
XXXIV.	Impresion de Campos Numericos y Expresiones	---	58
XXXV.	Los Agregados COUNT-MIN-MAX-PERCENT-TOTAL-AVERAGE		59
XXXVI.	El Estatuto LET	---	59
XXXVII.	El Estatuto IF-THEN-ELSE	---	59
XXXVIII.	Ejemplos	---	60

7. PERFORM

I.	Introduccion	---	61
II.	Definicion de los Formatos de Pantalla	---	61
III.	Uso de los Comandos QUERY, NEXT, y PREVIOUS	---	66
IV.	Cambios en el Contenido de la Base de Datos	---	66
V.	Actividades Avanzadas	---	67
VI.	Ejemplos	---	68

8. ANEXO

Pasos para usar la base de datos INFORMIX

RESUMEN EN ESPANOL DE LOS MANUALES DE LA BASE DE DATOS

****** INTRODUCCION DE INFORMIX ******

I. Que es un Sistema Manejador de la Base de Datos?

Desde el punto de vista del usuario de computadoras, un sistema manejador de la base de datos provee facil acceso interactivo a un almacen de informacion. Ademas, esta informacion es mantenido por el sistema en un formato bien definido desde el cual los reportes impresos son facilmente obtenidos.

II. Que es un Sistema de Base de Datos Relacional?

La informacion almacenada en computadoras es usualmente dividida en grupos, (que comunmente se les denomina archivos (files)), el poder obtener informacion combinada (es decir de mas de un archivo), para un reporte en particular, se le llama base de datos relacional, para ello es necesario que los grupos tengan un elemento en comun.

Para cada archivo de la base de datos, se forman:

archivo.dat	para los datos
archivo.idx	para los indices

Para la base de datos, hay:

archivo.dbd	contiene informacion del diccionario de datos

El sistema manejador de la base de datos INFORMIX, es una coleccion de programas de computadora que permiten al usuario crear un sistema completo de manejo de datos, a menudo sin utilizar programacion. Cuando es requerida la programacion, se puede hacer por medio de Report-Writer ya que es facil y rapido.

Una vez que se tiene disenado el formato de la base de datos, se hace uso de DBBUILD para construir la base de datos, sus archivos y campos. Aqui es donde se crean los archivos siguientes:

Archivo.dat
Archivo.idx
Archivo.dbd (contiene la informacion acerca de la estructura de la base de datos)

El numero de archivos esta limitado solamente por la cantidad de me

moria principal disponible.

Tres programas de INFORMIX:

MASTER MENU
ENTER2
PERFORM

solo pueden ser usados en terminales que tienen pantalla.

INFORMIX puede ser ejecutado tecleando

informix

y aparecera

1.	Perform	Data Entry and query by forms
2.	Acego	Run Ace Report
3.	Informer	Query Lenguaje
4.	Selection	Database Selection
5.	Dbstatus	Database Administration
6.	Schema	Print Database Schema
7.	Enter2	Screen-Oriented data entry
8.	Enter1	Data Entry
9.	System	Operating System Command

en este momento se puede seleccionar cualquier opcion, tecleando el numero correspondiente.

Para terminar la sesion, teclear la letra "b".

Cualquier comando dado a un programa ineractivo de INFORMIX, puede ser terminado presionando la tecla DEL.

El usuario de la base de datos normalmente conoce cuales campos son de importancia para efectuar la mayoria de las busquedas. En el archivo STORES (que se vera adelante), los campos CNAME y ONAME son los principales, ya que sobre ellos se efectuaran la mayoria de las busquedas. Para que la busqueda sea mas rapida es necesario tener los campos mas importantes indexados (con indices). Los indices pueden ser especificados cuando se construye la base de datos con el esquema DBBUILD.

DBSTATUS puede ser usado para adicionar indices a un archivo, no hay limite para el numero de campos indexados.

Cuando una llave es creada, el usuario puede especificar si va a contener valores duplicados o ser llave unica. En la base de datos STORES, CNAME (nombre del cliente) debe ser llave unica, puesto que en el archivo CUSTOMERS no debe haber clientes duplicados (con el mismo nombre), sin embargo en el archivo ORDERS, ONAME (nombre del cliente) puede estar mas de una vez, puesto que un mismo cliente puede tener varios pedidos colocados.

Si una falla del sistema ocurre, el respaldo en cinta puede ser usado para restaurar la informacion, la cual estara actualizada hasta el ultimo respaldo que se haya efectuado, y las transacciones posteriores se pueden recobrar del audit trail por medio del comando RECOVER de

DBSTATUS.

DBSTATUS tiene los comandos LOAD y UNLOAD que permiten cargar y descargar datos en los archivos de la base de datos.

DBBUILD es usado para crear la base de datos y puede ser usado para cambiar la estructura de la base de datos, adicionando, borrando, acrecentar o acortar campos.

INFORMER es un lenguaje interactivo, con el que se puede imprimir un archivo o parte del mismo. Puesto que su sintaxis esta basada sobre el modelo de base de datos relacional, puede proporcionar respuestas a preguntas que impliquen la relacion de mas de un archivo.

ACE REPORT WRITER es similar a INFORMER, pero con mas flexibilidad en el formateo de datos. Las operaciones aritmeticas pueden ser realizadas con ACE y tambien algunas operaciones agregadas tales como TOTAL, PERCENT, COUNT y AVERAGE.

RESUMEN EN ESPANOL DE LA BASE DE DATOS INFORMIX

******* D B B U I L D *******

I. Introduccion.

DBBUILD es un compilador para un lenguaje que permite al usuario - crear una base de datos nueva, adicionarle archivos, o cambiar el formato de archivos existentes.

Un programa DBBUILD debe ser escrito y compilado para cada archivo de la base de datos. El programa especifica el nombre de la base de datos que contendra el archivo, el nombre del archivo y los nombres de -- los campos en el archivo.

Cuando la especificacion del primer archivo de una base de datos - es compilada, la base de datos sera creada.

DBBUILD creara 2 o 3 archivos del sistema operativo. Cada base de- datos tiene un archivo del diccionario de datos. Si el archivo del sistema describe el primer archivo de una base de datos, entonces DBBUILD- creara un archivo del diccionario de datos. Este archivo es nombrado -- usando el nombre completo de la base de datos o truncado a 10 caracte-- res, y la terminacion ".dbd", para el diccionario de la base de datos.

Cada archivo de datos de la base de datos es representado por 2 -- archivos, un archivo que contiene los datos y otro que contiene los - - indices. La terminacion para el archivo de datos es ".dat" y para el de indices es ".idx". No hay limite del numero de indices que este archivo pueda tener.

Solamente los primeros 10 caracteres del nombre de la base de da-- tos y archivos son considerados por DBBUILD.

Cada nombre de campo o archivo en la base de datos debe ser unico.

Los nombres de campos y archivos, deben tener al menos 2 caracte-- res de longitud.

El numero de archivos y campos permitidos es limitada por el - - - direccionamiento de la memoria principal de un proceso en la computadora, la cual es usualmente una limitacion del sistema operativo. El maximo tamano de un campo es 2048 bytes (caracteres). Esto es tambien el -- maximo tamano de un registro.

En los ejemplos de este manual, se tomara la siguiente base de datos.

II. Base de Datos STORES

customer

cname	address	balance
Brooks,B.	7 Apple Rd.	10.50
Field,W.	43 Cherry Ln.	0
Robin,R.	12 Heather Ct.	23.45
Hart,W.	65 Lark Rd.	43.00
Court,S.	56 Blossom Rd.	0
English,D.	82 Alpine Rd.	0

orders

oname	item	quantity
Brooks,B.	Work Bench	5
Brooks,B.	Saw	1
Robin,R.	Work Bench	3
Hart,W.	File	3
Robin,R.	Hammer	8
Court,S.	Saw	3
Court,S.	File	5
English,D.	File	1
English,D.	Hammer	2

Las especificaciones para los 2 archivos de esta base de datos son como sigue:

```
database stores

file customers

field cname    type character      length 15
field address  type character      length 15
field balance  type double

end
```

```
database stores

file orders

field oname    type character      length 15
field item     type character      length 15
field quantity type integer

end
```

III. Definiciones de DBBUILD

III.A Identificadores

Identificadores, tal como el nombre del archivo y nombres de campos pueden ser compuestos de cualquier combinacion de letras, numeros y la- raya que esta en la tecla DEL. Sin embargo, los identificadores deben - empezar con una letra. Un identificador puede tener hasta 50 caracteres de longitud y deben tener al menos 2 caracteres. Todos los caracteres - son significativos en comparaciones.

III.B Constantes.

La constante que es usada en DBBUILD es la entera, es usada para - describir la longitud de los campos alfabeticos.

III.C Comentarios.

Los comentarios pueden ser incluidos en cualquier parte en el - - texto, estos deben estar entre los parentesis ( ).

III.D Llaves.

Las llaves pueden ser una combinacion de mayusculas y minusculas. Sus usos son como sigue:

A. DATABASE database_name

DATABASE nombra la base de datos para la cual el programa de - - DBBUILD es escrito. La palabra DATABASE junto con el nombre de la base de datos son obligatorios.

B. FILE file_name

FILE nombra el archivo para el cual el programa DBBUILD es escrito La palabra FILE junto con el nombre del archivo son obligatorios. El -- nombre del archivo debe ser unico dentro de la base de datos.

C. FIELD field_name

La palabra FIELD es requerida para cada campo definido en el ar-- chivo. Los nombres de los campos pueden tener hsta 50 caracteres de -- longitud. El nombre del campo debe ser unico en el archivo y en la ba- se de datos.

D. TYPE CHARACTER.

El tipo CHARACTER, le dice al compilador que el campo es una se-- cuencia de caracteres. Si la frase TYPE CHARACTER es usada, debe de -- especificarse la longitud. TYPE CHARACTER es abreviada como TYPE CHAR.

E. LENGTH 10.

Esta especificacion de longitud indica que un campo tipo alfanumerico tiene una longitud de 10 caracteres. Los campos pueden tener de 1 a 2048 caracteres de longitud. La palabra LENGTH puede ser abreviada como LEN.

F. TYPE INTEGER.

El tipo INTEGER le dice al compilador, que el campo es entero. --- INTEGER puede ser abreviada como INT. Los enteros consumen 2 bytes de espacio (la misma cantidad de espacio que 2 caracteres). Los numeros enteros no pueden tener parte decimal y deben estar entre el rango --- de -32,768 a +32,767.

G. TYPE LONG.

El tipo LONG, consume 4 bytes de espacio. Esto permite la utilizacion de enteros en el rango de -2,147,483,648 a +2,147,483,647.

H. TYPE FLOAT.

En este tipo son permitidos 7 digitos significativos. Consume 4 bytes de espacio en el registro.

I. TYPE DOUBLE.

El tipo DOUBLE, le dice al compilador que el campo es un numero de punto flotante. 14 digitos significativos son permitidos. Consume 8 --- bytes de espacio en el registro.

J. TYPE SERIAL.

Los campos de datos tipo SERIAL son asignados a numeros de series secuenciales. Si no es especificado el numero inicial, al primer registro le sera asignado el numero 1. Solamente un campo de este tipo es --- declarado en un archivo de la base de datos. Si un tipo SERIAL es declarado, el sera la llave primaria para el archivo, a menos que la llave primaria sea especificada.

Ejemplo

```
field empno     type serial  start 10000 (inicia en 10001)
field deptno    type serial              (incia en 1)
```

K. TYPE DATE.

Los datos introducidos en los campos de tipo DATE, seran esperados en la forma 'mm/dd/yy'. Las variantes disponibles son EDATE 'dd/mm/yy' y YDATE 'yy/mm/dd'. Aqui 'mm' es el mes, 'dd' el dia y 'yy' es el a/no.

L. TYPE MONEY.

Los campos de tipo MONEY son almacenados como numeros de doble precision de punto flotante.

M. TYPE COMPOSITE.

Los campos COMPOSITE combinan 2 o mas (maximo 8) campos definidos previamente. La palabra COMPOSITE puede ser abreviada como COMP.

Ejemplo

```
database inventory

file stock

field aisle    type integer
field shelf    type integer
field position type composite aisle, shelf
```

IV. Indices.

Los indices para los campos pueden ser creados al mismo tiempo que la base de datos es creada, especificando "index" en el esquema.

Ejemplo

```
field cname    type  character length 25 index
```

El campo indexado sera considerado como un identificador unico a menos que la especificacion del indice tenga "dups" indicando que duplicidad de entrada es permitida.

V. Llaves Primarias.

Una llave primaria es un identificador unico en el registro, como por ejemplo la ficha puede ser una llave primaria, identificando el registro del trabajador.

Cualquier tipo de dato, incluyendo los compuestos, pueden ser usados como llaves primarias.

Solamente una llave primaria puede ser asignada en un archivo de la base de datos.

Si un tipo "serial" es declarado y no es especificada la llave primaria, el "serial" sera la llave primaria.

Ejmeplo

```
field empno    type serial index primary (no se permite
                                          duplicidad)
field edeptno  type integer index dups    (se permite
```

field edeptmn type integer index duplicidad> {no se permite duplicidad>

VI. Ejemplo de una Corrida

El esquema DUMMY es creado por el usuario usando el editor del sistema operativo. Una vez que el archivo (custschem) ha sido escrito, el compilador es corrido tecleando el siguiente comando:

dbbuild custschem

Si la compilacion es satisfactoria, los archivos del sistema operativo para el archivo descrito en el esquema seran constuidos. Para cada archivo de la base de datos, se crearan al menos 2 archivos del sistema operativo. Ellos seran filename.idx y filename.dat (donde filename es el nombre especificado en el archivo del esquema).

Si la compilacion no fue satisfactoria, entonces el compilador imprimira mensajes de errror y creara un archivo que contenga una lista de errores, esencialmente, el archivo del esquema con el numero de error y mensajes sobre el problema. Este archivo sera designado con el nombre del archivo del esquema y la terminacion ".err".

VII. Cambios a la Estructura de una Base de Datos.

Se requieren basicamente 2 etapas.

En la primera, el archivo del esquema que fue usado para crear la base de datos debe ser editado (meterse a editor) para efectuar los cambios deseados.

En la segunda, es recompilado el archivo cambiado, por medio del siguiente comando

dbbuild -r nombre del archivo del esquema
---

aqui se preguntara si se desea continuar con el proceso para el cambio, con lo que se debe contestar "y" (si se desea continuar) o "n" (si no se desea continuar).

******* E N T E R 2 *******

I, Introduccion.

ENTER2 se puede correr sin la necesidad de definir y formatear -- pantallas. Permite adicionar, suprimir y actualizar registros de cualquiera de los archivos de una base de datos, estos cambios se realizan registro por registro.

La base de datos REALTY, la cual tiene los archivos LISTINGS y -- AGENTS, sera usada a manera de ejemplo.

Los esquemas para estos 2 archivos son:

```
database realty

file listings

field listing_number        type integer                    index
field address               type character   length 15
field city                  type character   length 10
field footage               type integer
field bedrooms              type integer
field baths                 type integer
field lot_size              type double                  (acres)
field corner_lot            type character    length 1 ("y" o "n")
field schools               type character    length 1 (escuela  )
                                                       (cerca   )
                                                       ("y" o "n")
field asking_price          type money
field sales_price           type money
field broker                type character    length 15

field listing_note1         type character    length 45
field listing_note2         type character    length 45
field listing_note3         type character    length 45

end


database realty

file agents

field agent_number          type integer                    index
field agent_name            type character    length 15
field agent_address         type character    length 20
field agent_city            type character    length 10
field agent_zip_code        type long
field agent_phone           type character    length 8
field agent_salary          type money

end
```

II, Ejecucion de ENTER2.

ENTER2 puede ser corrido tecleando el siguiente comando

enter2

En seguida se preguntara por el nombre de la base de datos que es- accesada y luego el nombre del archivo que va a ser manejado primero. - Si solo hay un archivo, este es accesado automaticamente.

Otra forma de correr ENTER2 es por medio del menu principal de - - INFORMIX, para lo cual se teclea el numero que le corresponde.

Despues que ha sido seleccionado el archivo, aparecera en la panta lla el esqueleto con los conceptos del registro.

Ejemplo

Seleccionando el archivo LISTINGS, aparecera en la pantalla lo siguiente:

Commands: Find Next Previous Update Add Delete Select Redraw Bye?

```
listing_number      [       ]
address             [                ]
city                [            ]
footage             [       ]
bedrooms            [       ]
baths               [       ]
lot_size            [                ]
corner_lot          [ ]
schools             [ ]
asking_price        [                ]
sales_price         [                ]
broker              [                ]
listing_note1       [                                                 ]
listing_note2       [                                                 ]
listing_note3       [                                                 ]
```

Cualquiera de los comandos puede ser ejecutado tecleando la primer letra de su nombre. Aquellos comandos que ocurren en mas de una etapa,- pueden ser abortados presionando la tecla DEL.

El comando ADD es usado para adicionar nuevos registros al archivo.

El comando FIND es usado para buscar y desplegar un registro basa- do en el contenido de uno de sus campos.

Una vez que un registro es desplegado en la pantalla, se puede eli minar por medio del comando DELETE o modificar por medio del comando -- UPDATE.

El comando NEXT y PREVIOUS localizan registros siguientes y ante- riores respectivamente al registro que esta en la pantalla.

Si se desea cambiar de base de datos, archivo o campo, se puede --

por medio del comando SELECT.

El comando REDRAW es usado para eliminar basura que es generada por diversas causas, alterando el contenido de la informacion.

El comando BYE termina la sesion de ENTER2.

III. Hechos Sobresalientes.

1. Checa entradas en forma interactiva.

2. Realiza busquedas aun cuando solamente uno o mas de los caracteres del valor de la llave son conocidos.

3. Mantiene la integridad de los datos en un medio ambiente de multi-usuarios.

IV. Chequeo de Datos Interactivamente.

A traves de DBBUILD, la longitud y tipo del campo son puestos en el diccionario de datos. ENTER2 tiene acceso a esta informacion y checa el dato que es introducido por el usuario cuando ejecuta los comandos ADD (adicion) y UPDATE (actualizacion). Si la entrada no corresponde con el tipo de dato introducido, se desplegara un mensaje de error.

V. Busquedas Genericas.

Por medio del uso del comando FIND, el usuario puede localizar un registro basado en el contenido de uno de sus campos. Si el campo es de tipo CHARACTER, entonces el usuario puede especificar solamente los primeros caracteres. Una busqueda basada sobre una parte de un campo, es conocida como busqueda generica.

VI. Los Comandos y sus Usos.

VI.A El comando FIND.

El comando FIND, permite al usuario localizar registros basandose en el valor de un campo. Una vez que se ha localizado y desplegado en la terminal, el usuario puede modificarlo usando el comando UPDATE o eliminarlo del archivo usando el comando DELETE.

El comando FIND es ejecutado presionando la tecla F, con lo cual aparece el esqueleto del archivo seleccionado anteriormente y un encabezado con 4 opciones.

```
FIND: Value First Last Same?

listing_number      [      ]
address             [                ]
city                [          ]
footage             [      ]
```

```
bedrooms          [       ]
baths             [       ]
lot_size          [                ]
corner_lot        [ ]
schools           [ ]
asking_price      [                ]
sales_price       [                ]
broker            [                ]
listing_note1     [                                  ]
listing_note2     [                                  ]
listing_note3     [                                  ]
```

VALUE.

El comando FIND es a menudo usado para localizar un registro basado en el valor (VALUE) de uno de sus campos. En este caso, el usuario debe teclear una V, luego debe teclear el valor que sera usado para la busqueda. Una vez que el registro es encontrado sera desplegado en la terminal.

FIRST.

Se despliega el primer registro del archivo.
Para hacer uso de esta opcion se debe teclear la letra F.

LAST.

Se despliega el ultimo registro del archivo.
Para hacer uso de esta opcion se debe teclear la letra L.

SAME.

Es para continuar una busqueda sobre campos no indexados, para encontrar registros subsecuentes con el mismo valor de la busqueda.
Para hacer uso de esta opcion se debe teclear la letra S.

VI.B El comando NEXT.

Cuando un registro es encontrado usando una busqueda basada sobre un campo indexado, entonces el uso del comando NEXT desplegara cualquier registro que tenga el mismo valor de la busqueda. Si no hay registros que tengan el mismo valor de la busqueda, NEXT desplegara el siguiente registro mayor al ultimo desplegado.

Si NEXT es usado cuando el campo presente no es indexado, desplegara otro registro basandose en el orden cronologico con que fueron introducidos o en la llave primaria.

VI.C El comando PREVIOUS.

Este comando es el complemento del comando NEXT. Desplegara el registro anterior al que esta desplegado en la pantalla.

VI.D El comando UPDATE.

Antes que un registro pueda ser cambiado, debe ser encontrado primero con el comando FIND, NEXT o PREVIOUS. Entonces el comando UPDATE puede ser usado tecleando la letra U.

Siempre que el cursor pasa por el ultimo campo, se preguntara si el registro esta terminado, con lo que se contestara 'y' en caso de que lo este o en caso contrario presionar la tecla RETURN, con lo que el cursor se posisionara en la parte superior del registro.

Si el archivo que se esta manejando con ENTER2 tiene mas campos que los que aparecen en la pantalla, cuando se pasa por el ultimo campo apareceran los siguientes campos faltantes.

La tecla DEL puede ser usada en cualquier punto para terminar prematuramente el comando que se esta ejecutando.

VI.E El comando ADD.

Este comando es ejecutado tecleando la letra A, y es usado para dar de alta registros.

VI.F El comando DELETE.

Una vez que se tiene el registro desplegado en la pantalla, puede ser eliminado con este comando, tecleando la letra D y contestando con una 'y' a la pregunta de:

delete?

VI.G El comando SELECT.

Este comando es usado para cambiar de campo, pagina, archivo o base de datos.

Este comando es ejecutado tecleando la letra S.

VI.H El comando REDRAW.

Por varias causas se puede generar basura en la pantalla, alterando el contenido de la misma, para eliminarla es necesario presionar la letra R.

VI.I El comando BYE.

Este comando es la entrada final en una sesion de DBSTATUS. Origi-

na que se cierren las bases de datos y archivos que fueron abiertos durante la sesion de DBSTATUS y termina los procesos interactivos.

RESUMEN EN ESPANOL DE LOS MANUALES DE LA BASE DE DATOS

***** I N F O R M E R *****

I. Introduccion.

INFORMER permite al usuario, examinar un archivo o partes especificas del mismo, asimismo se pueden formular preguntas que impliquen mas de un archivo, generando los reportes que se deseen.

Con la finalidad de mostrar los reportes que pueden ser generados, consideraremos el siguiente ejemplo:

Base de datos STORES (tienda)

customers (archivo de clientes)

cname(nombre cliente)	address(direccion)	balance
Brooks,B.	7AppleRd.	10.5
Field,W.	43CherryLn.	0
Robin,R.	12HeatherCt.	23.45
Hart,W.	65LarkRd.	43.00
Court,S.	56BlossomRd.	0
English,D.	82AlpineRd.	0

orders (archivo de ordenes o pedidos)

oname(nombre cliente)	item(articulo)	quantity(cantidad)
Brooks,B.	WorkBench	5
Brooks,B.	Saw	1
Robin,R.	WorkBench	3
Hart,W.	File	3
Robin,R.	Hammer	8
Court,S.	Saw	3
Court,S.	File	5
English,D.	File	1
English,D.	Hammer	2

Como se observa en el archivo de clientes, se encuentra la informacion relativa al cliente, es decir su nombre, direccion y si tiene algun pago vencido (balance). En el archivo de pedidos, se encuentran los pedidos que han realizado los clientes, y que aun estan pendientes de surtir.

II. Sintesis y estructura

II.A. Identificadores

Los identificadores de INFORMER son los nombres de campos y archivos que fueron definidos usando DBBUILD.

Los archivos creados con DBBUILD son considerados permanentes.

Los archivos temporales pueden ser creados durante una sesion y pueden contener hasta 200 campos y se puede usar mas de uno durante la sesion. Los archivos temporales pueden hacerse permanentes haciendo uso del comando UNLOAD.

II.B. Comandos

Los comandos pueden extenderse mas de una linea, y deben ser terminados con la palabra 'end' o con ';', excepto los comandos 'bye' y 'help'.

III. Comando y Definiciones.

INFORMER puede ser ejecutado por medio del menu de opciones al ejecutar INFORMIX, seleccionando la opcion 3, o directamente tecleando 'informer', en cualquiera de estas dos opciones, INFORMER preguntara por la base de datos con la que se desee trabajar, en este punto se puede proporcionar el nombre de la base de datos o teclear 'b' (bye) para finalizar la sesion. Despues que la base de datos a sido seleccionada satisfactoriamente, el usuario puede ejecutar cualquier comando, los cuales se veran posteriormente. Cualquier comando puede ser terminado prematuramente presionando la tecla 'DEL'. INFORMER solo leera datos de la base de datos.

III.A. El comando 'PRINT'

Por medio de este comando, es posible obtener informacion de los archivos, ya sea una parte de los registros que cumplan cierta(s) condicion(es) deseada(s), o determinados campos del archivo, ejemplo:

```
> print cname balance where blance = 0 end
```

El resultado que se obtiene es el siguiente:

```
cname                balance

Field, W.              0.00
Court, S.              0.00
English, D.            0.00
```

Otro ejemplo, donde se seleccionaran los clientes que hayan pagado sus facturas o que su deuda no exceda de 20 pesos, es el siguiente:

```
 > print
>> cname balance
>> where balance >= 0.0 and balance <= 20.00
>> end
```

El resultado es el siguiente:

```
cname                balance

Field, W.              0.00
Court, S.              0.00
```

English, D. 0.00
Brooks, B. 10.50

Para abundar sobre el comando PRINT ver hojas 10 a 15 del manual de - INFORMER.

Sintaxis del comando PRINT.

```
                    /                                  \
                   |   WITHOUT HEADINGS                 |
                   |   TO "filename"                    |
> print fieldlist  <    TO PIPE "programname"           > end o ;
                   |   whereclause                      |
                   |   joinclause                       |
                    \                                  /
```

Donde las opciones entre las llaves pueden ser usadas todas o ninguna y en cualquier orden.

fieldlist.- uno o mas nombres de archivos o campos separados por - -- comas "," o blancos " " o la palabra "end".

joinclause.- la frase "from join on" o la palabra "joining" seguida - por:

campo = campo o campo = opcional campo

whereclause.- la palabra "where" seguida por:

= , <> , < , > , <= , >= , matches

y combinada con "and" , "or" o "not"

cuando se usa el "and" y "or", primero se ejecuta el -- "and" y luego el "or".

III.B. El comando "READ"

El comando READ es usado para crear archivos temporales de la base de datos, los cuales pueden ser impresos o usados en posteriores estatutos - de lectura o impresion, ejemplo:

```
> read into x cname balance where balance = 0;

> print x;
```

Los resultados son:

cname	balance
Field, W.	0.00
Court, S.	0.00
English, D.	0.00

En este ejemplo "x" es un archivo temporal, el cual se usara en el siguiente ejemplo:

```
> read into y x.cname oname item
> joining x.cname = oname;

> print y;
```

Los resultados son:

cname	oname	item
Court, S.	Court, S.	Saw
Court, S.	Court, S.	File
English, D.	English, D.	File
English, D.	English, D.	Hammer

La sintaxis usada para el comando PRINT , puede ser usada para el comando READ, excepto las frases WITHOUT, HEADINGS y TO FILENAME. Los archivos temporales son borrados por el sistema al finalizar la sesion.

III.C. El comando UNLOAD

La estructura tipica de un comando UNLOAD es la siguiente:

```
> unload tmp to "filename" end
```

donde: tmp, es el nombre del archivo temporal
filename, es el nombre del archivo que tendra la informacion de tmp.

Usando el comando UNLOAD de INFORMER, datos pueden ser preparados para cargarlos en nuevos archivos de la base de datos. Esto es a menudo util para inicializar nuevos archivos de la base de datos, con datos que son derivados de otras partes de la base de datos.

III.D. El comando ASSIGN

El comando ASSIGN y sus clausulas UNION, MINUS e INTERSECT permiten al usuario ejecutar todas las operaciones basicas del algebra relacional. Por medio de estas clausulas el usuario puede crear la union, interseccion o diferencia entre dos archivos temporales, ejemplos:

```
> assign c = a union b end

> assign c = a intersect b end

> assign c = a minus b end
```

assign puede ser abreviado como "ass".

Para llevar a cabo estas operaciones, los archivos deben ser compatibles, es decir los campos deben ser del mismo tipo y longitud y en el mismo orden. El archivo especificado antes de las clausulas, determinara el nombre de los campos en el nuevo archivo. En los ejemplos anteriores, los campos en el archivo "c" seran identificados por los nombres de los

campos en el archivo "a". Los archivos que resultan de estos comandos no- tendran campos duplicados.

La clausula UNION, genera un archivo temporal con los registros de -- los dos archivos unidos, omitiendo registros duplicados.

La clausula INTERSECT, genera un archivo temporal con los registros - que son comunes en los dos archivos considerados.

La clausula MINUS, genera un archivo temporal con los registros que -- son la diferencia entre los dos archivos considerados.

IV. El estatuto EXECUTE

```
> execute file-name
```

Este comando puede ser usado en un programa, de tal forma que cuando- INFORMER es corrido, automaticamente se ejecutaran los comandos estable-- cidos en el mencionado programa, ejemplo:

```
informer stores billspaid
```

En esta instruccion, "stores" es el nombre de la base de datos y - -- "billspaid" el nombre del archivo que contiene la serie de instrucciones- que deseamos se ejecuten, ejemplo:

```
> execute billspaid end

 read into x cname balance where balance = 0 end

> print x end
```

El resultado es el siguiente:

```
cname                balance

Field, W.              0.00
Court, S.              0.00
English, D.            0.00
```

El comando execute puede ser abreviado como "ex".

IV.F. El comando HELP

INFORMER tiene la facilidad de que en cualquier momento durante un -- proceso normal, se puede hacer uso del comando HELP para conocer la sin-- taxis o significado de cualquier comando de INFORMER. El usuario solo - - tiene que teclear "help" y dar "RETURN", y todos los comandos de INFORMER seran desplegados.

IV.G. El comando BYE

Este comando, cierra los archivos accesados para la sesion, elimina - los archivos temporales y termina la sesion de INFORMER. Este comando es- ejecutado introduciendo la palabra "bye" o "b" y tecleando "RETURN".

> bye

el resultado sera:

program over

V. Sintaxis

Los comandos PRINT, UNLOAD y READ son muy similares, las pequenas -- diferencias son: el estatuto PRINT entrega los resultados directamente a- la pantalla, el estatuto READ prepara un archivo temporal, el cual permanece disponible durante la sesion y el estatuto UNLOAD prepara un archivo permanente en el formato de "carga" de archivos de INFORMIX.

Sintaxis de los estatutos READ, PRINT y UNLOAD

```
                          /                          \
                         |   WITHOUT HEADINGS         |
  READ INTO x fieldlist  |   TO "filename"            |
> PRINT fieldlist        <   TO PIPE "programname"    > end o ;
  UNLOAD fieldlist       |   whereclause              |
                         |   joinclause               |
                          \                          /
```

Las opciones entre las llaves pueden ser usadas en cualquier orden y- puede ser desde ninguna hasta todas.

fieldlist. Uno o mas nombres de archivos o campos separados por - - - comas "," o blancos " " o la palabra "end".

joinclause. La frase "from join on" o la palabra "joining" seguida -- por:

campo = campo o campo = optional campo

whereclause. La palabra where seguida por cualquiera de los siguien-- tes operadores

= , <> , <= , >= , matches

y combinada con "and" , "or" o "not"

cuando se use el "and" y "or", primero se ejecuta el - - "and" y luego el "or".

V.B. Clasificacion de la salida

INFORMER permite clasificar las respuestas de las preguntas estable-- cidas por medio de la clausula SORT. La salida puede ser clasificada - -- sobre uno o mas campos incluidos en la pregunta. Hasta 8 campos pueden -- ser usados en una clasificacion jerarquica. La clausula SORT debe apare-- cer al final del ultimo estatuto read o print.

La clasificacion de los campos puede ser en orden ascendente o descendente. El default es ascendente. Los campos especificados por el SORT deben ser indexados, a menos que los datos deseados, sean leidos primero en un archivo temporal y entonces clasificados.

V.C. Creacion de alianzas

El JOIN es comunmente usado en INFORMIX, para hacer enlaces entre los archivos, sin embargo hay necesidad en algunos casos de hacer enlaces entre un mismo archivo, para lo cual se hace uso de la clausula ALIAS para el archivo en cuestion, ejemplp:

```
alias cus = customer;
print cust_num, employer, salary, co_app,
     cus.employer,cus.salary
     joining co_app = cus.cust_num;
```

Esto dara numeros, empleos y salarios de los dos solicitantes de un mismo prestamo, si deseamos que se proporcione informacion de aquellos prestamos que solo tienen un solicitante (no tiene co_app), simplemente se anade la palabra "optional" antes de cus.cust_num en la linea donde esta el "joining".

Campos etiquetados

Puede en algunos casos ser util renombrar campos. Estos nuevos nombres son listados en la linea de los comandos READ o PRINT, ejemplo:

```
alias cus = customer;
print cust_num, employer, salary, co_app,
     coapp_emp = cus.employer,
     coapp_sal = cus.salary
     joining co_app = optional cus.cust.num;
```

Esto dara cust_num (numero del cliente), employer (empleo) y salary (salario) para todos los clientes y su co_applicants en caso de que tengo

V.D. Enlaces opcionales

Permiten la seleccion de registros, aun cuando no correspondan en el archivo enlazado, ejemplo:

```
> print cname item joining cname = oname
>> where balance = 0 end
```

El resultado es el siguiente:

```
cname              item

Court, S.          Saw
Court, S.          File
English, D.        File
English, D.        Hammer
```

En este caso, aparecieron los registros que cumplen la condicion de que su balance sea cero y ademas que se cumpla la condicion de enlace entre los dos archivos. Si se desea que aparezcan todos los registros que

tengan su blance igual a cero, aun cuando no exista enlace entre los dos- archivos para algunos archivos, es decir se hace opcional el enlace, - -- ejemplo:

```
> print cname item joining cname = optional oname
>> where blance = 0 end
```

El resultado es el siguiente:

```
cname                item

Field, W.
Court, S.            Saw
Court, S.            File
English, D.          File
English, D.          Hammer
```

V.E. Enlaces multiples

Ver hoja 33 del manual de INFORMER

V.F. Expresiones aritmeticas

Las expresiones aritmeticas pueden ser usadas en clausulas WHERE, --- campos temporales o en asignacion de variables.

Las clausulas WHERE permiten seleccion de registros, donde la selec-- cion esta basada sobre un valor aritmetico de mas de dos campos, tal - -- como:

```
print file where field1*field2 > 100;
```

Una variable puede ser asignada usando el comando LET, y la variable- puede ser impresa o usada en la clausula WHERE, ejemplo:

```
let hisal = average of salary + 5000;
print hisal, cust_num, salary where salary > hisal;
```

la variable "average" se explicara en la siguiente seccion.

V.G. Variables agregadas

INFORMER proporciona las siguientes variables agregadas:

COUNT , AVERAGE , TOTAL , MIN y MAX

AVERAGE puede ser abreviado como AVG

El agregado deseado puede ser asignado usando el comando LET, impre- so usando PRINT o leido en un archivo temporal usando READ INTO.

Los estatutos READ y PRINT, no pueden tener agregados y otros campos- en el mismo estatuto. En caso de que se desee, una variable con la informacion agregada debe ser creada con el estatuto READ e impresa con otros- campos en un estatuto PRINT, ejemplo:

```
print count of loan_num;
let av = average of loan_num;
read into rmin min of rate;
```

Sintaxis

```
                  /          Ñ
                 |    COUNT   |
                 |    TOTAL   |
LET variable = <  |  AVERAGE   | > of expresion;
                 |    MIN     |
                 |    MAX     |
                  Ñ          /
```

```
                  /          Ñ
                 |    COUNT   |
                 |    TOTAL   |
       PRINT    <  |  AVERAGE   | > of expresion;
                 |    MIN     |
                 |    MAX     |
                  Ñ          /
```

```
                              /          Ñ
                             |    COUNT   |
                             |    TOTAL   |
READ into archivo_temporal  <  |  AVERAGE   | > of expresion;
                             |    MIN     |
                             |    MAX     |
                              Ñ          /
```

V.II. El comando PROMPT

Usando el comando PROMPT, INFORMER permite que un programa al ser ejecutado, solicite informacion, tal como una fecha, nombre .etc. ejemplo

```
prompt for variable_temporal using " Introduzca el nombre deseado' ";
```

V.I. Actualizaciones, adiciones y bajas en un gran numero de registros.

Ver hojas 39 y 40 del manual de INFORMER

Para mas informacion consultar el manual de INFORMER

RESUMEN EN ESPANOL DE LOS MANUALES DE LA BASE DE DATOS

***** D B S T A T U S *****

I. Introduccion.

DBSTATUS es una herramienta diseñada para el programador de la base de datos y para el administrador de la base de datos (DBA).

Usando DBSTATUS, el DBA puede anadir o borrar indices en archivos o imprimir un esquema de archivo seleccionado o estatus. Archivos o base - de datos completas pueden ser borradas.

Los datos no solamente son introducidos en la base de datos - - - - INFORMIX por medio de los programas ENTER1 y ENTER2, sino que teniendo- los en un archivo del sistema operativo, DBSTATUS puede cargar los datos de este archivo a uno de la base de datos. DBSTATUS puede usarse para -- pasar datos de la base de datos INFORMIX a un archivo del sistema operativo.

La siguiente base de datos se tomara como ejemplo:

Base de datos STORES (tienda)

customers (archivo de clientes)

cname(nombre cliente)	address(direccion)	balance
Brooks,B.	7AppleRd.	10.5
Field,W.	43CherryLn.	0
Robin,R.	12HeatherCt.	23.45
Hart,W.	65LarkRd.	43.00
Court,S.	5&BlossomRd.	0
English,D	82AlpineRd.	0

orders (archivo de ordenes o pedidos)

oname(nombre cliente)	item(articulo)	quantity(cantidad)
Brooks,B.	WorkBench	5
Brooks,B.	Saw	1
Robin,R.	WorkBench	3
Hart,W.	File	3
Robin,R.	Hammer	8
Court,S.	Saw	3
Court,S.	File	5
English,D.	File	1
English,D.	Hammer	2

Como se observa, en el archivo de clientes se encuentra la informacion relativa al cliente, es decir; su nombre, direccion y si tiene algun pago vencido (balance). En el archivo de pedidos, se encuentran los pedidos que han realizado los clientes, y que aun estan pendientes de surtir.

II. Adicion y Eliminacion de Indices

Una de las consideraciones mas importantes en el funcionamiento del sistema, es la decision de cuales campos tendran indices y cuales no. Las busquedas basadas sobre campos que estan indexados son mucho mas rapidas que las busquedas basadas sobre campos que no tienen, ya que en este caso se genera un archivo y la busqueda es secuencial. A pesar de la rapidez de acceso con campos indexados, no es recomendable tenerlos inecesariamente, ya que ocupan mayor espacio de disco y se consume mas tiempo cada vez que un registro es adicionado o eliminado, debido a que los indices de los archivos deben ser actualizados y rebalanceados.

III. Control de Cambios de un Archivo (audit trail control)

La dinamica habilitacion y deshabilitacion del audit trail, es posible por medio de los comandos START AUDIT y STOP AUDIT respectivamente.

IV. Eliminacion de Archivos y Base de Datos.

Por medio de los comandos ERASE FILE y ERASE DATABASE es posible eliminar archivos y base de datos respectivamente.

V. Reproduccion del Esquema de la Base de Datos.

Un esquema de archivo dado, puede ser reproducido del diccionario de datos via el comando PRINT STATUS de DBSTATUS. Este comando puede ser usado tambien para imprimir el esquema de la base de datos completa.

VI. Definiciones del Comando DBSTATUS.

Una sesion de DBSTATUS se inicia corriendo el programa DBSTATUS. De la misma forma que otros componentes de INFORMIX, puede ser corrido desde el MASTER MENU, en cuyo caso la base de datos presente sera asumida. Si no es especificada la base de datos al correr DBSTATUS, se preguntara por la base de datos que sera procesada, ejemplo;

```
dbstatus

please select database

>stores

The database 'stores' has been selected successfully
```

Si una base de datos diferente es accesada durante una sesion de DBSTATUS, entonces el comando SELECT DATABASE es usado para cambiar de base de datos.

DBSTATUS acepta comandos ya sea en letras mayusculas o minusculas, y cualquier comando puede ser terminado presionando la tecla DEL.

El usuario puede salirse de DBSTATUS tecleando la letra "b" y presionando "RETURN".

VII. El Comando PRINT SCHEMA

>PRINT SCHEMA [FOR file-name] [TO [PIPE] "file or program"]

El comando PRINT SCHEMA despliega el esquema del archivo seleccionado, incluyendo informacion de los cambios del archivo (audit trail).

Si un archivo no es especificado, entonces el esquema de la base de datos completa es desplegado, ejemplo:

```
> print schema

PRINT SCHEMA

database stores

file customers                          permission read update add

field cname          type char 15
field address        type char 15
field balance        type money         permission read update


file orders                             permission all

field oname          type char 15
field item           type char 15
field quantity       type integer


> print schema for customers

PRINT SCHEMA

database stores

file customers                          permission read update add

field cname          type char 15
field address        type char 15
field balance        type money         permission read update
```

VIII. El comando PRINT STATUS

> PRINT STATUS [FOR file-name] [TO [PIPE] "file or program"]

El comando PRINT STATUS, despliega informacion general acerca del archivo, incluyendo la longitud de los registros, el numero de registros y la existencia y estado de indices y audit trails.

La version simple PRINT STATUS, mostrara la informacion para todos los archivos de la base de datos. Cuando solo se desea informacion de un archivo, este debe ser especificado. La salida sera en la terminal, a menos que la clausula "TO" sea usada. Usando la clausula "TO", la salida puede ser enviada a un archivo del sistema operativo o a la impresora usando la opcion "PIPE".

```
> print status

PRINT STATUS

database stores

file customers

data record length       38
number of records         6
number of indexes         2
balance              duplicates allowed
cname                duplicates not allowed


file orders

data record length       32
number of records         9
number of indexes         1
oname                duplicates allowed

> print status for orders

PRINT STATUS

database stores

file orders

data record length       32
number of records         9
number of indexes         1
oname                duplicates allowed
```

IX. El comando ADD INDEX

> ADD INDEX FOR field-name

o

> ADD INDEX FOR field-name ALLOWING DUPS

En los comandos ADD INDEX y DELETE INDEX, una llave puede ser definida en cualquier campo de cualquier tipo. El campo sera considerado - - como identificador unico a menos que la clausula ALLOWING DUPS sea adi-- cionada al comando. Si el campo a ser indexado es del tipo alfanumerico, no puede ser mayor a 118 caracteres.

```
> print status por customers

PRINT STATUS

database stores

file customers

data record length     30
number of records       6
number of indexes       0

> add index for cname


ADD INDEX

pausa mientras el nuevo indice es formado

un indice ha sido adicionado al campo "cname".
```

X. El comando DELETE INDEX

> DELETE INDEX FOR field-cname

El campo que es especificado en el comando DELETE INDEX, debe tener un indice. Llaves primarias no pueden ser borradas en DBSTATUS.

```
> print status for customers

PRINT STATUS

database stores

file customers

data record length      38
number of records        6
number of indexes        2
cname                duplicates not allowed
```

```
balance                 duplicates allowed

> delete index for cname
```

DELETE INDEX

El indice del campo 'cname' ha sido borrado.

```
> print status for customers

PRINT STATUS

database stores

file customers

data record length       38
number of records         6
number of indexes         1
balance                 duplicates allowed
```

XI. Los Comandos ERASE DATABASE, FILE o AUDIT TRAIL

```
> ERASE DATABASE database-name

          o

> ERASE FILE file-name

          o

> ERASE AUDIT FOR file-name
```

El comando ERASE DATABASE borra todos los archivos de la base de datos, ejemplo:

```
> erase database stores
```

ERASE DATABASE

La base de datos que fue borrada, fue la que estaba presente. Favor de seleccionar una nueva base de datos. Se puede teclear 'bye' para salirse del programa.

```
please select database.

> bye
```

El comando ERASE FILE borra los datos del archivo, cambios del archivo (audit trail), indices y elimina el archivo seleccionado del diccionario de datos.

> erase file orders

ERASE FILE

El archivo 'orders' ha sido borrado.

El comando ERASE AUDIT TRAIL borra la historia de los cambios - - - (audit trail) del archivo especificado. El archivo permanecera, solo su historia sera removida.

> erase audit trail customers

ERASE AUDIT TRAIL

El archivo de la historia de los cambios (custaudit) ha sido borrado.

XII. El Comando RENAME FIELD,

> RENAME FIELD old-field-name TO new-field-name

El comando RENAME cambia el nombre de un campo

> rename field cname to customername

RENAME FIELD

El campo 'cname' ha sido renombrado 'customername'

Este cambio puede corroborarse con el comando "print schema".

XIII. El comando START AUDIT

> START AUDIT FOR file-name TO "path-name"

El comando START AUDIT, deja en posicion de encendido el switch del audit trail en el diccionario de datos para el archivo especificado.

> start audit for customers to "custaudit"

Con el comando "print status" se verifica si el audit esta encendido o apagado.

XIV. El comando STOP AUDIT

> STOP AUDIT FOR FILE file-name

El comando STOP AUTIT, deja en posicion de apagado el switch del -- audit trail en el diccionario de datos para el archivo especificado.

> stop audit for file customers

Con el comando "print status" se verifica si el audit esta encendido o apagado.

XV. El Comando RECOVER FILE

> RECOVER FILE file-name

El comando RECOVER FILE debe ser usado para restaurar un archivo a- partir de su audit trail en el caso de que el sistema falle.

> recover file customers

RECOVER FILE

pausa mientras el archivo es recuperado.

El archivo "customers" de la base de datos ha sido recuperado del - archivo "custaudit". Una vez que el archivo ha sido recuperado, el audit trail debe ser borrado y creado nuevamentte con START AUDIT.

XVI. El comando LOAD.

> LOAD file-name FROM [PIPE] "file or program"

El comando LOAD asume que el contenido del archivo es solamente datos y que cada registro es de longitud fija.

La opcion PIPE puede ser usada para cargar datos, los cuales son -- generados por otro programa, en lugar de leerlos de un archivo.

> load customers from "custdata"

Si los datos estan siendo cargados de un dispositivo remoto, tal -- como un manejador de cintas, el comando es escrito como sigue:

> load customers from pipe "/dev/rmt0/custdata"

XVII. El comando UNLOAD

> UNLOAD file-name TO [PIPE] "file or program"

El comando UNLOAD copiara el archivo file-name a un archivo secuencial del sistema operativo.

Si todos los campos del archivo son de tipo alfanumerico, la des- - carga del archivo puede ser la impresora.

> unload customers to "custdata"

Si los datos estan siendo descargados a un dispositivo remoto, tal-como un manejador de cintas, el comando es:

> unload custtomers to pipe "/dev/rmt0/cusdata"

XVIII. El Comando SELECT DATABASE

> SELECT DATABASE database-name

El comando SELECT DATABASE, es usado para cambiar de base de datos. Cuando el programa DBSTATUS es corrido por primera vez, se le preguntara al usuario por el nombre de la base de datos de interes, si no ha sido especificada antes. Usando el comando SELECT DATABASE, el usuario puede cambiar de base de datos sin correr nuevamente el programa DBSTATUS.

> select database stores

La palabra "database" puede ser abreviada como "db"

> select db realty

XIX. El Comando HELP

> HELP

Este comando es usado para obtener una lista de los comandos disponibles en DBSTATUS.

> help

HELP

Comands:

ADD INDEX FOR field-name [ALLOWING DUPS]
BYE
DELETE INDEX FOR field-name
ERASE AUDIT FOR file-name
ERASE DATABASE database-name
ERASE FILE file-name
HELP
LOAD file-name FROM [PIPE] "regular-file-name"
PRINT SCHEMA [FOR file-name] [TO [PIPE] "regular-file-name"]

```
PRINT STATUS [FOR file-name] [TO [PIPE] 'regular-file-name']
RECOVER FILE file-name
RENAME FIELD old-field-name TO new-field-name
SELECT DATABASE database-name
START AUDIT FOR file-name TO 'audit-trail-name'
STOP AUDIT FOR file-name
UNLOAD file-name TO [PIPE] 'regular-file-name'
```

XX. El Comando BYE

Este comando es para terminar la sesion de DBSTATUS. Esto origina que el programa de DBSTATUS cierre la base de datos y archivos que fueron abiertos durante la sesion.

```
> bye

BYE


Program over.
```

RESUMEN EN ESPANOL DE LOS MANUALES DE LA BASE DE DATOS

******* R E P O R T W R I T E R *******

I. Introduccion. Que es un Report Writer?

ACE es un lenguaje de programacion disenado especifica-- mente par la produccion de reportes a partir de una base de datos.

Que es un Relational Report Writer?

Consiste en poder relacionar la informacion que se tiene en diferentes archivos que forman la base de datos, para generar los reportes deseados, para lo cual es necesario que los archivos tengan en comun el contenido de un campo.

Hay 3 operaciones que ACE puede ejecutar para permitir - especificaciones sencillas de la informacion que es im-- presa en el reporte, que son:

seleccion
proyeccion
union o enlace

Seleccion y Proyeccion operan solamente sobre un archivo Seleccion es la operacion que selecciona un subconjunto- de registros que seran incluidos en el reporte. Proyec-- cion es usada para seleccionar campos del archivo, los - cuales son deseados en el reporte. Union o Enlace es --- ejecutado sobre 2 archivos.

Un report writer, permite agrupar a un mismo tiempo par-- tes de una base de datos, usando las operaciones de - -- Seleccion, Proyeccion y Enlace.

II. Ejempos de reportes

IIA. Haciendo uso de la base de datos "stores" la cual tiene los archivos "orders" y "customers", tenemos:

```
rep1
database stores end
read orders end
format every record end
```

```
aceprep rep1
acego rep1
```

Resultados:

oname	item	quantity
Brooks, B.	Work Bench	5
Brooks, B.	Saw	1
Robin, R.	Work Bench	3
Hart, R.	File	3
Robin, R.	Hammer	8
Court, S.	Saw	3
Court, S.	File	5
English, D.	File	1
English, D.	Hammer	2

El programa tiene 3 comandos solamente:

```
DATABASE
READ
FORMAT
```

Los comandos de ACE deben tener al menos estos 3 comandos

IIB. Ejemplo de seleccion de campos de un archivo.

Solamente se desea en el reporte los campos de ITEM y QUANTITY

```
rep2
database stores end
read
    item quantity
end

format every record end
```

```
aceprep rep2
acego rep2
```

Resultados

item	quantity
Work Bench	5
Saw	1
Work Bench	3
File	3
Hammer	8
Saw	3
File	5
File	1
Hammer	2

II.C Seleccion de aquellos registros que cumplen cierta condicion.

Se desea leer los campos ITEM y QUANTITY en donde QUANTITY sea mayor a 2.

```
rep3
database stores end
read
     item quantity
     where quantity>2
end

format every record end

acego rep3
```

Resultados

item	quantity
Work Bench	5
Work Bench	3
File	3
Hammer	8
Saw	3
File	5

III. Clasificacion de la Salida

Para que la informacion reportada se obtenga en una clasificacion deseada, se hace uso del comando "SORT".

Ejemplo. El reporte se desea clasificar por articulo y cantidad (ITEM y QUANTITY)

```
rep4
database stores end
read into x orders end
sort by item,quantity end

format every record end
```

Resultados

oname	item	quantity
English, D.	File	1
Hart, W.	File	3
Court, S.	File	5
English, D.	Hammer	2
Robin, R.	Hammer	8
Brooks, B.	Saw	1
Court, S.	Saw	3
Robin, R.	Work Bench	3
Brooks, B.	Work Bench	5

Cuando se efectua una clasificacion, es necesario crear un archivo temporal (en el ejemplo anterior el archivo x), haciendo uso de la clau-

sula "into". El comando "SORT" admite hasta 8 campos para clasificarlos, y se puede especificar si la clasificacion es ascendente o descendente, (por default se tiene clasificacion ascendente).

Ejemplo

```
sort by item quantity descending end
```

aqui la clasificacion para "item" sera ascendente (por default) y para "quantity" sera en orden descendente.

IV. El Comando Format

El comando "format" se usa para formatear los reportes.
Este comando, debe tener al menos una clausula, y a su vez las clausulas tienen estautos.

Ejemplo: Se usan 2 clausulas "PAGE HEADER" y "ON EVERY RECORD"

```
rep5
database stores end
read orders end
sort by item,quantity end

format
      page header
         print " Sorted and Grouped Item List"
         skip 2 lines
      on every record
         print item,8 spaces,quantity

end


ocego rep5
```

Resultados

```
   Sorted and Grouped Item List

   File                    1
   File                    3
   File                    5
   Hammer                  2
   Hammer                  8
   Saw                     1
   Saw                     3
   Work Bench              3
   Work Bench              5
```

La clausula "PAGE HEADER" es para indicar el encabezado, con el estatuto "PRINT" se imprime el encabezado deseado, y con el estauto "SKIP" se ponen lineas en blanco entre el encabezado y el cuerpo del reporte. Los estautos en la clausula "ON EVERY RECORD" imprimen el cuerpo del reporte.

Las clausulas que permiten imprimir informacion en los lugares - - deseados de un reporte son las siguientes:

```
PAGE HEADER
FIRST PAGE HEADER
ON LAST RECORD
PAGE TRAILER
ON EVERY RECORD
BEFORE GROUP OF field
AFTER GROUP OF field
```

PAGE HEADER. El encabezado sale igual en todas las hojas del repor te.

FIRST PAGE HEADER. La informacion especificada solo sale en la pri mera hoja del reporte.

ON LAST RECORD. La informacion deseada aparece despues del ultimo- registro procesado.

PAGE TRAILER. La informacion deseada aparece en la parte de abajo- de las paginas del reporte.

ON EVERY RECORD. La informacion deseada aparece en cada registro - procesado.

BEFORE y AFTER GROUP of field. La informacion especificada aparece antes o despues de cada grupo, para el campo especificado.

Los estatutos que pueden ser usados en estas clausulas son:

```
print
skip n lines
skip to top of page
pause "prompt definido por el usuario"
if then
if then else
let variabele = expresion
while exp do estatuto
for var = exp to exp [step exp] do estatuto
call croutine(lista de parametros)
croutine(lista de parametros)
```

V. Calculos.

El siguiente ejemplo muestra el estatuto "PRINT" con la sintaxis - "USING". El reporte es el mismo que el del ejemplo anterior, excepto -- que se adiciona la clausula "ON LAST RECORD". El estatuto "PRINT" que - se encuentra despues de "ON LAST RECORD", imprimira el total del campo- "QUANTITY" considerando todos los registros, para lo cual el agregado - "TOTAL" es usado.

```
repó
database stores end
read into x all of orders where all end
sort by item,quantity end
```

```
format
      page header
           print "Sorted and Grouped Item List"
           skip 2 lines
      on every record
           print item, 8 spaces, quantity
      on last record
           skip 2 lines
           print "There where a total of",
                 total of (quantity) using "<,<<<"
                 print "items ordered."

end
```

acego rep6

Resultados

```
              Sorted and Grouped Item List
              File                      1
              File                      3
              File                      5
              Hammer                    2
              Hammer                    8
              Saw                       1
              Saw                       3
              Work Bench                3
              Work Bench                5
```

VI. Agrupacion de datos en el reporte

Haciendo uso de las clausulas BEFORE GROUP OF y AFTER GROUP OF, es posible dejar lineas en blanco indicando separacion de grupo, imprimir subtotales o subencabezados.

Ejemplo

```
rep7
database stores end
read into x all of orders where all end
sort by item,quantity end

format
      page header
           print "Sorted and Grouped Item List"
           skip 2 lines
      on every record
           print item, 8 spaces, quantity
      after group of item
           skip 1 line
      on last record
           skip 2 lines
           print "There were a total of",
           total of (quantity) using "#,###"
```

```
            print "items ordered."
    end

    acego -q rep7
```

Con el "-q" se suprimen los mensajes que han estado apareciendo en los ejemplos anteriores (omitidos en este manual, pero que el usuario - vera en la terminal al ejecutar cualquier ejemplo anterior).

Resultados

```
            Stored and Grouped Item List

            File                    1
            File                    3
            File                    5

            Hammer                  2
            Hammer                  8

            Saw                     1
            Saw                     3

            Work Bench              3
            Work Bench              5



            There were a total of  31
            items ordered.
```

En el ejemplo anterior, la clausula AFTER GROUP OF, solo se uso -- para dejar un espacio entre grupo de articulos, sin embargo, se puede - usar para otras cosas, tal como imprimir subtotales a los grupos de - - articulos.

Ejemplo

```
        rep8
        database stores end
        read into x all of orders where all end
        sort by item,quantity end
        format
             page header
                 print "Sorted and Grouped Item List"
                 skip 2 lines
             on every record
                 print item, 8 spaces,quantity
             after group of item
                 skip 1 line
                 print "There were a total of ",
                       group total of (quantity) using "##,###"
                 print item without trailing blanks,
                       "items ordered."
                 skip 1 line
```

```
        on last record
            skip 2 lines
            print "There were a total of ",
                  total of (quantity) using "#,###"
                  print "items ordered."

    end

    acego rep8
```

Resultados

```
    Stored and Grouped Item List


    File                          1
    File                          3
    File                          5

    There were a total of         9
    File items ordered.

    Hammer                        2
    Hammer                        8

    There were a total of        10
    Hammer items ordered.

    Saw                           1
    Saw                           3

    There were a total of         4
    Saw items ordered.

    Work Bench                    3
    Work Bench                    5

    There were a total of         8
    Work Bench items ordered.


    There were a total of        31
    items ordered.
```

VII. Uso de datos de 2 archivos.

Los reportes generados pueden tener informacion de mas de un archivo.

Ejemplo

```
rep9
database stores end
output
  left margin 0
  right margin 80
end
```

```
read
  customers item quantity
  from join on cname = oname
end

format every record end

acego rep9
```

Resultados

cname	address	balance	items	quantity
Brooks, B.	7 Apple Rd.	10.10	Work Bench	5
Brooks, B.	7 Apple Rd.	10.10	Saw	1
Robin, R.	12 Heather Ct.	23.45	Work Bench	3
Robin, R.	12 Heather Ct.	23.45	Hammer	8
Hart, W.	65 Lark Rd.	43.00	File	3
Court, S.	56 Blossom Rd.	0.00	Saw	3
Court, S.	56 Blossom Rd.	0.00	File	5
English, D.	82 Alpine Rd.	0.00	File	1
English, D.	82 Alpine Rd.	0.00	Hammer	2

VIII. Como Escribir y Ejecutar los Programas ACE

Preparacion de un programa ACE para ejecucion.

ACE es compuesto de 2 programas, ACEPREP y ACEGO.

El programa ACEPREP, traduce el programa ACE y crea un archivo que tiene ".arc" como los ultimos 4 caracteres del nombre del archivo. El archivo ".arc" (Ace Report Control), contiene las tablas de control del reporte ACE, el uso de ACEPREP es de la siguiente manera:

```
aceprep "nombre del archivo"
```

Cuando este comando es ejecutado y el proceso no es correcto, se crea un archivo con los errores, cuya terminacion es ".err", y en caso de no haber errores, la compilacion es satisfactoria y se crea el archivo con la erminacion ".arc", en cuyo caso se esta en condiciones de generar el reporte por medio del comando

```
acego "nombre del archivo"
```

Pasos para la generacion del archivo ACE.

1. Entrar a editor especificando un nombre al archivo

Ejemplo

```
i ordersrep
```

2. Teclear el programa ACE.

```
database stores end
read into x oname, item where all end
sort by oname end

format
  page header
      print "Sorted and Grouped Name List"
      skip 2 lines
  on every record
      print oname, 2 spaces, item
  after group of name
      print

end
```

3. Grabar el programa (salvarlo) presionando la tecla CTRL y sin soltarla presionar una sola vez la tecla DEL

4. Ejecutar ACEPREP con el siguiente comando

```
aceprep ordersrep
```

IX. Ejecucion de un programa ACE

Una vez que el programa es compilado correctamente, la ejecucion puede hacerse por medio del siguiente comando

```
acego ordersrep
```

Resultados

```
        Sorted and Grouped Name List

Brooks, B.      Work Bench
Brooks, B.      Saw

Court, S.       Saw
Court, S.       File

English, D.     File
English, D.     Hammer

Hart, W.        File

Robin, R.       Work Bench
Robin, R.       Hammer
```

X. Impresion de la Fecha y Hora

La fecha y hora de la ejecucion de un reporte puede ser impresa usando DATE y TIME en el estatuto Print

```
print "La fecha de la corrida de este reporte"
print "es ",date,"."
skip 2 lines
print "La hora de la corrida de este reporte"
print "es ",time,"."
```

Resultados

```
La fecha de la corrida de este reporte
es Mon Aug 10, 1983
   ^
   |_____________ fecha en que se efectuo la corrida


La hora de la corrida de este reporte
es 10:14:44
   ^
   |_____________ hora en que se efectuo la corrrida
```

XI. Reportes Usando Parametros

A menudo hay reportes que tienen la misma informacion y el mismo - formato de salida, tal como las ordenes colocadas para un articulo en - particular, para lo cual se alimenta el nombre del articulo (parametro) y se obtiene el reporte de las ordenes que hay para ese articulo.

Programa

```
cat itemrep1
database stores end
define
      param[1] selecart type character length 20
end
read orderes
      while item = secart
end

format every record end
```

El reporte anterior (itemrep1), es ejecutado selectivamente para - el articulo deseado. Ejemplo, si el articulo es "File", la ejecucion es de la siguiente forma:

```
acego itemrep1 File
```

en caso de que el nombre del articulo sea de mas de una palabra, el - - siguiente comando debera ser usado

```
acego itemrep1 "Work Bench"
```

Ejecutando el programa anterior con

```
acego itemrep1 File
```

los resultados son los siguientes

oname	item	quantity
Hart, W.	File	3
Court, S.	File	5
English, D.	File	1

Algunos reportes necesitan mas de un parametro

Cuando esto sucede, hay que especificar los parametros deseados en la seccion de "define" del programa.

Ejemplo. Un programa para seleccionar los registros del archivo -- "orders", bajo 2 parametros (articulo y cantidad) es el siguiente:

```
cat itemrep2
database stores end
define
    param[1] selecart type character length 20
    param[2] seleccant type integer
end
read
  into x
  all of orders
  where item = selecart
        and quantity > seleccant
end

format every record end
```

El comando "READ" usa los parametros "item" y "quantity" para seleccionar los registros de un articulo en particular y que tiene una cantidad mayor a cierto valor. Ejecutando el programa por medio del siguiente -- comando

```
acego -q itemrep2 File 2
```

en el cual el articulo deseado fue "File" y la cantidad "2", tenemos -- los siguientes resultados

oname	item	quantity
Hart, W.	File	3
Court, S.	File	5

XII. Uso de variables para subcalculos

El ejemplo siguiente, usara una variable para guardar el subtotal- de el numero de articulos ordenados y se imprimiran cada vez que se cambia de articulo.

```
cat runningtot
database stores end
define
  variable runningtotal type integer
```

```
end
read into x all of orders where all end
sort by item, quantity end

format
   first page header
       let runningtotal = 0
   page header
       print "Sorted and grouped Item List"
       skip 2 lines
   on every record
       print item, 8 spaces, quantity
       let runningtotal = runningtotal + quantity
   after group of item
       skip 1 line
       print "There were a total of ",
            group total of (quantity) using "#,###"
       print item without trailing blanks,
              " items ordered."
       print "This totals ",
              runningtotal using "#,###",
              " thus far."
       skip 1 line
   on last record
       skip 2 lines
       print "There were a total of ",
              total of (quantity) using "#,###"
              print "items ordered."
end


acego -q runningtotal
```

Resultados

```
  Sorted and Grouped Item List

  File                          1
  File                          3
  File                          5

  There were a total of         9
  File items ordered.
  This totals      9 thus far.

  Hammer                        2
  Hammer                        8

  There were a total of        10
  Hammer items ordered.
  This totals     19 thus far.

  Saw                           1
  Saw                           3

  There were a total of         4
```

```
Saw items ordered.
This totals    23 thus far.

Work Bench                    3
Work Bench                    5

There were a total of         8
Work Bench items orderes.
This totals    31 thus far.


There were a total of        31
items ordered.
```

La variable runningtotal fue definida en el comando "define" como una variable entera. Se pueden definir hasta 100 variables.

Se utilizo el comando "let", para hacer la asignacion de la cantidad a la variable runningtotal.

El estatuto "let" puede tener cualquier expresion numerica del lado derecho del signo "=".

XIII. Control de la impresion con el estatuo IF-THEN-ELSE

En un estatuto IF-THEN-ELSE, el numero de estatutos entre el BEGIN y END es ilimitado. Se puede omitir la parte ELSE. Pueden estar anidados, es decir un IF-THEN-ELSE, puede controlar otros IF-THEN-ELSE.

Si estatutos IF-THEN-ELSE que contienen estatutos de impresion son usados en las clausulas PAGE HEADER o PAGE TRAILER, se debe tener cuidado de que la parte IF y la parte ELSE, tengan el mismo numero de lineas. En otras clausulas puede ser usado sin esta consideracion.

XIV. Pausa en la presentacion de la informacion

El estatuto PAUSE es usado para hacer una pausa en la presentacion de la informacion al ser obtenida esta.

Ejemplo

```
pause "Teclee return para continuar el proceso"
```

El estatuto PAUSE, solo opera cuando el reporte es impreso en una salida estandar (pantalla).

XV. Impresion con el WITHOUT TRAILING BLANKS

Cuando se especifica la longitud de un campo para introducir informacion, y no es llenado, tal como sucede en la especificacion de nombres de personas, el espacio restante es llenado con blancos, y al ser impresos estos nombres, los espacios en blanco crean una presentacion inadecuada, tal como:

```
field firstname      type character length 20
```

```
field middleinitial type character length 1
field lastname      type character length 20
```

Siendo estos campos impresos usando el estatuto regular PRINT, los resultados son los siguientes:

```
print firstname, " ",middleinitial, ". ", lastname

John                J. Smith
```

Cuando los campos son impresos usando el estatuto PRINT con la - - opcion WITHOUT TRAILING BLANKS, los resultados son los siguientes:

```
print firstname without trailing blanks,
      " ", middleinitial, ". ",lastname

John J. Smith
```

La frase WITHOUT TRAILING BLANKS, puede ser sutituido por la palabra CLIPPED, con el mismo efecto.

```
print firstname clipped 1 space middleinitial
      ". " lastname
```

XVI. Salto a la parte superior de la pagina

Esto se acostumbra usar en reportes clasificados. Por ejemplo si - en el archivo "orders", hubiera gran cantidad de registros, y fueran - impresos clasificados por "item", se pueden requerir varias hojas para- imprimir un articulo en particular, y es conveniente que para cada articulo diferente, su inicio de impresion fuera en el comienzo de una nueva hoja.

Para obtener cambio de hoja entre grupos de registros diferentes,- el estatuto "SKIP TO TOP OF PAGE" debe ser usado dentro de la clausula- AFTER GROUP OF, tal como se muestra

```
after group of item
      skip to top of page
```

El estatuto SKIP TO TOP OF PAGE, no es permitido en las clausulas- PAGE HEADER y PAGE TRAILER.

XVII. Definicion del lenguaje ACE

Comentarios

Con la finalidad de que los programas puedan ser analizados y comprendidos por otros programadores de ACE, es necesario ponerles mensajes. Estos mensajes son llamados comentarios y van entre {}.

Ejemplo:

```
database stores end
define
   param[1] thisitem  {Este parametro sera usado para}
   type character     {retener un articulo seleccionado}
```

```
        length 20
    end
    read
        orders
        where item = thisitem
    end

    {Note que el parametro param[1] fue usado arriba y asigna
     do a la variable "thisitem". Solamente los registros se-
     leccionados seran ahora disponibles para el proceso.
     }

    format every record end
```

XVIII. Numeros

Los numeros que son usados en programas ACE pueden ser formados con o sin punto decimal.

Los numeros enteros y los reales (los de punto decimal), pueden - tener un signo positivo o negativo enfrente de ellos. El signo de dollar o coma no son permitidos (son permitidos solo en el PRINT USING).

Los numeros reales pueden usar exponenciacion, por ejemplo el nume- ro 345,000,000,000,000 es mejor representarlo como 345e12. No debe haber espacios entre los elementos de este numero. El numero base y el exponen te pueden tener signo positivo o negativo. Los numeros en notacion - --- exponencial pueden tener porcion decimal.

XIX. Estructura de identificadores y variables

Identificadores tales como el nombre de la base de datos, nombres - de archivos y nombres de campo que son usados en programas ACE deben se- guir las mismas reglas que fueron usadas cuando fueron construidos en -- DBBUILD.

Los nombres de las variables que son usadas en programas siguen las mismas reglas como los nombres de campo en especificaciones DBBUILD. - - Ellas pueden tener una longitud de hasta 50 caracteres y todos ellos son considerados significativos cuando son hechas comparaciones.

Todos los caracteres alfabeticos y numericos pueden ser usados para los nombres de las variables, en adicion al caracter especial '_' de la- tecla DEL.

XX. El comando Database

Este comando es la parte mas simple de un programa ACE. Especifica- el nombre de la base de datos de la cual el reporte va ha ser escrito. - Este comando es requerido (no opcional).

Ejemplo

```
        database stores end
```

XXI. El comando define

Este comando es opcional. La sintaxis para definir una funcion es -

como sigue:

```
define function predict end
```

Este estatuto DEFINE alerta al compilador ACEGO que una funcion - - llamada "predict" puede ser llamada posteriormente en el comando "format"

Enseguida vamos un comando DEFINE creando 3 variables para usar en- un programa ACE

```
define
   variable counter type integer
   variable specialtotal type double
   variable selectionvalue type character length 10
end
```

La primer variablee (counter) es entera. Tendra inicialmente un - valor de cero, y tomara valores de acuerdo a la ejecucion de los estatu- tos "let" o "for". Si expresiones son asignadas a esta variable, la - -- parte fraccional del valor sera ignorada.

La segunda variable es declarada numerica de doble presicion y de - punto flotante. Esta variable puede recibir valores numericos con por-- ciones decimales, sin perder la parte fraccional.

La tercer variable es declarada alfanumerica de longitud 10.

Hasta 100 variables pueden ser definidas en un programa ACE.

XXII. El comando INPUT

Este comando es usado solamente para especificar que algunas va--- riables que han sido definidas en el comando DEFINE son inicializadas - cuando el programa esta corriendo, preguntando al usuario en forma in-- teractiva por el valor inicial.

La sintaxis para este comando es la siguiente

```
input
   prompt for selectionvalue using
   "Please select the item for the report > "
end
```

Cuando el reporte inicia su corrida, lo que se le asigno en el es- tatuto "prompt" sera desplegado en la terminal, en este caso

```
Please select the item for the report
```

La respuesta del usuario a la terminal, sera usada para inicializar el- valor de la variable.

El comando INPUT es opcional.

XXIII. El comando OUTPUT

Hay varios valores de default que son asumidos por ACE. Por ejem--

plo, los margenes superior e inferior son 3 lineas, el margen izquierdo son 5 espacios, la longitud de la pagina son 66 lineas , el ancho son 132 caracteres y finalmente el reporte por default sera desplegado en la terminal que corrio el reporte.

Cualquiera de estos valores de default, pueden ser cambiados por medio del comando opcional OUTPUT. Se ofrece a continuacion un comando OUTPUT que cambia estos valores de default:

```
output
  page length 24
  right margin 80
  left margin 0
  top margin 2
  bottom margin 1
  report to "screenfile"
end
```

Este comando OUTPUT escribira el reporte en un archivo

Si la salida se desea directamente en una impresora, la siguiente sintaxis es usada

```
output
  report to print
end
```

El comando OUTPUT es opcional.

XXIV. El comando READ

ACE requiere al menos un comando READ, pero permite cualquier numero.

Hay una forma especial del comando READ la cual efectua lecturas optimizadas de un archivo. La sintaxis es la siguiente:

```
read all of orders end

o

read orders end
```

La restriccion que tiene esta lectura optimizada, es que debe ser el ultimo estatuto de lectura en el reporte.

Esta forma de lectura, no crea un archivo temporal, con lo que se ejecuta mas rapido y ademas no requiere espacio de disco para area de trabajo.

Sintaxis del comando READ

```
> READ [INTO X] fieldlist  (whereclause)    end o ;
                           (joinclause)
```

fieldlist. Uno o mas nombres de archivos o campos separdos por ---
comas "," o blancos " " o la palabra "end".

joinclause. La frase "from join on" o la palabra "joining" seguida por:

campo = campo

whereclause. La palabra where seguida por cualquiera de los si----
quientes operadores

=, <>, <=, >=

y combinada con "and" , "or" o "not"

cuando se use el "and" y "or", primero se ejecuta el "and" y luego el "or".

XXV. El comando SORT.

El SORT es un comando que es usado para clasificar informacion antes de procesarla con el comando FORMAT. La sintaxis es:

sort by item quantity end

Esta clasificacion es primero por item y luego por quantity. La --
clasificacion por default es ascendente.

Hasta 8 campos pueden ser clasificados en el comando SORT.

Este comando es opcional.

Sintaxis

SORT BY XLISTA END

XLISTA = X[ASCENDING|DESCENDING]

o

XLISTA SEPARDOR X

X = Xnombre
o Xnombre[n,m]
donde n y m > 0
y m > n

Separador = una coma, la palabra end o uno o mas blancos o nuevas-
lineas.

XXVI. El comando FORMAT.

Este comando es compuesto de combinaciones de hasta 7 clausulas, -
que son:

first page header

```
page header
page trailer
on every record
before group of
after group of
on last record
```

XXVII. La clausula FIRST PAGE HEADER.

La informacion que esta bajo esta clausula solo sera impresa como encabezado en la primer hoja del reporte.

XXVIII. La clausula PAGE HEADER.

La informacion que esta bajo esta clausula sera impresa como encabezado en las hojas del reporte.

XXIX. La clausula ON EVERY RECORD.

Esta clausula es usada para procesar las lineas de detalle del reporte. Usualmente parte o todos los registros que fueron calificados por el ultimo estatuto READ son impresos por estatutos de impresion contenidos en esta clausula.

Esta clausula opcional.

XXX. Las clausulas BEFORE GROUP OF y AFTER GROUP OF.

Estas clausulas controlaran los estatutos de impresion que seran ejecutados, ya sea antes del cambio de grupo o despues del cambio del grupo, dependiendo de la clausula usada, la sintaxis usada es:

```
before group of state

before group of city

after proup of xxxx
```

XXXI. La clausula ON LAST RECORD.

Despues que el ultimo registro ha sido procesado, entonces las impresiones de ON LAST RECORD ocurriran, esto es muy util para impresiones de tablas.

XXXII. La clausula PAGE TRAILER.

La informacion condicionada bajo esta clausula sera puesta en la parte inferior de cada hoja del reporte.

Si el numero de pagina es deseado, un estatuto de impresion. "PRINT PAGENO" es usado en esta clausula y automaticamente apareceran las paginas numeradas. Ejemplo:

```
page trailer
   print 20 spaces "page.", pageno using "###"
```

XXXIII. Impresion de archivos.

El estatuto PRINT puede ser usado para imprimir el texto de cualquier archivo del sistema operativo.

```
print file "filename"
```

XXXIV. Impresion de campos numericos y expresiones.

Uso del estatuto PRINT USING.

El formato de impresion puede consistir de cualquier combinacion de los siguientes caracteres:

< * & # , . ( ) - + $

Estos caracteres pueden ser agrupados en funcion de su uso de la siguiente forma:

Grupo 1 * & # <

para imprimir el contenido de un campo o expresion de diferentes formas.

Grupo 2 , .

la coma es para separar los numeros enteros y el punto para separar la fraccion decimal.

Grupo 3 - + ( )

el menos es signo negativo, el mas es signo positivo y la cantidad que aparece dentro de los parentesis significa que es negativa.

Grupo 4 $

signo de pesos.

Ejemplo: Si el campo balance tiene el valor 23485.23 y se desea imprimir, se puede hacer por medio de la siguiente instruccion

```
print "El balance es "
      balance using "##,###,##&.&&"
```

El resultado seria

```
El balance es $   23,485.23
```

Para mas informacion ver pagina 99 del manual de Report Writer

Cuando se usa el caracter *, los lugares que quedan vacios cuando el numero a imprimir es mas pequeno que el campo especificado, son llenados con asteriscos (*).

Cuando se usa el caracter &, los lugares que quedan vacios son llenados con ceros (0).

Cuando se usa el caracter #, los lugares que quedan vacios son llenados con espacios en blanco.

XXXV. Los agregados COUNT, MIN, MAX, PERCENT, TOTAL y AVERAGE.

COUNT, es el mas simple de todos.

Ejemplo

```
print "Fueron ", count using "#####",
      " asistencias a la"
print "reunion este mes."
print
print "Esperamos ", 2 * count using "#####",
      " asistencias a la"
print "reunion el proximo mes."
```

PERCENT. Es usado similarmente a COUNT. El COUNT y PERCENT son usados solamente en las clausulas de AFTER GROUP OF.

TOTAL Y AVERAGE. Requieren una expresion aritmetica para ejecutar- el total o promedio de los registros sobre los cuales su alcance es - - especificado.

Ejemplo

```
print "El total de ventas del mes son ",
      total of ventas using "$$,$$$,$$$.&&",
      "."
```

XXXVI. El estatuto LET.

Asignaciones a variables que han sido definidas en el comando - - - DEFINE, pueden ser hechas por medio del estatuto LET, de la siguiente - forma:

```
let var = var + 1
```

Cuando son asignadas expresiones a variables enteras, la parte - - fraccional es eliminada.

XXXVII. El estatuto IF-THEN-ELSE

Ejemplo 1

```
if (precio_venta > average of precio_venta)
then
  begin
  print "Arriba del promedio"
  end
```

Ejemplo 2

```
if (precio_venta > average of precio_venta)
then
  print "Arriba del promedio"
else
  print "Abajo del promedio"
```

Nota. BEGIN y END son requeridos solamente si hay mas de un estatuto siguiendo a THEN o ELSE.

XXXVIII. EJEMPLOS

Ver otros ejemplos a partir de la hoja numero 111 del manual de -- Report Writer.

RESUMEN EN ESPANOL DE LOS MANUALES DE LA BASE DE DATOS

***** P E R F O R M *****

I. Introduccion.

Sirve de interfase entre el usuario y los datos de una base de datos, permitiendo adicionar, borrar, encontrar y actualizar - registros.

Permite al usuario colocar los campos en la pantalla donde - - desee y con la leyenda deseado. Para situaciones en que los -- datos no caben en una sola pantalla, PERFORM permite pantallas multiples, las cuales son enlazadas y vistas en secuencia en - en la teerminal.

PERFORM permite construir pantallas con campos de mas de un -- archivo, haciendo uso de JOING, el cual enlaza los archivos.

PERFORM permite efectuar validaciones, a los campos se les - - puede definir chequeo de rangos, puede especificarse que el -- dato es requerido (no opcional), puede ser especificado algo por default (ejem. nombre).

PERFORM permite seleccionar un registro o grupo de registros - leyendolos de la base de datos, sujeto a la condicion de los - siguientes operadores (=,<,>,=< y =>) asi como a rangos de - valores. Cualquier registro encontrado, puede ser modificado o borrado si se desea. Nuevos registros pueden ser anadidos!

II. Definicion de los formatos de pantalla

II.A. Diseno de la forma

Editor es usado para describir una forma de pantalla, creando un -- archivo, ya que permite un formato simple. Este formato sera descrito en- esta seccion. Se usara un ejemplo de bienes raices. El esquema para la -- base de datos es el siguiente:

```
database realty <---------- realty, es el nombre de la base de datos (puede
                            ser cualquier nombre)

file listings  <----------- listings,nombre del archivo, que puede ser otro

field listing_number                type integer
field address                       type character  length 20
field city                          type character  length 20
field footage                       type integer
field bedrooms                      type integer
field baths                         type integer
field lot_size                      type double
field corner_lot                    type character  length 1
field schools                       type character  length 1
```

```
    field asking_price                      type money
    field sales_price                       type money
    field broker                            type character  length 15
    field listing_note1                     type character  length 135

    file agents <----------nombre del archivo, puede ser cualquiera

    field agent_number                      type integer
    field agent_name                        type character  length 15
    field agent_address                     type character  length 20
    field agent_city                        type character  length 10
    field agent_zip_code                    type long
    field agent_phone                       type character  length 8
    field agent_salary                      type money
```

Una pantalla de datos para este ejemplo seria:

```
    database realty

    screen
    {

            Lista de los bienes raices de las familias residentes
                        en Santa Clara California

  agent name [list1           ]

  listing number       street address                          city
    [list2 ]       [list3                  ]            [list4

    footage   bedrooms      baths      lotsize          corner   school
  [list5    ]  [list6 ]     [list7]   [list8        ]     [x]      [y]


             asking price                sale price
            [list9       ]              [list10      ]

  listing note [list11                                          ]

    }

end

    attributes

    list1 = broker;
    list2 = listing_number, include = (1 to 30000 ), required;
    list3 = address, required;
    list4 = city, required, default = "Santa Clara";
    list5 = footage, include = (800 to 4000),
          comments = "Enter a number from 800 to 4000.";
    list6 = bedrooms, include = (1 to 15),
          comments = "Enter a number from 1 to 15.";
    list7 = baths;
```

```
list8 = lot_size,
        comments = "Enter acres or portions; .5 is half an acre.";
x       = corner_lot, include =(Y,N,y,n),upshift;
y       = schools, include = (Y,N,y,n),upshift;
list9 = asking_price, required;
list10 = sales_price;
list11 = listing_note1[1,45];

end
```

La descripcion de la pantalla esta entre los caracteres (y), mas de una pantalla puede ser especificada en una forma. Las pantallas pueden tener datos de hasta 8 diferentes archivos.

Para definir la posicion de los campos se usan campos etiquetados dentro de los parentesis rectangulares, los nombres de estos campos pueden ser tan pequenos como un caracter, pero siempre deben empezar con una letra. Los nombres etiquetados son asociados con los nombres de los campos de la base de datos, en la seccion de atributos.

Si todos los caracteres en un campo de la base de datos se desplegaran en el campo de la forma, entonces el campo de la forma debera tener la misma longitud que la definida en el campo de la base de datos.

Si se tiene un campo alfanumerico en la forma, con menor longitud que el campo de la base de datos, solo se desplegara el campo parcialmente (se trunca hasta llenar la longitud especificada en la forma). En el caso de campos numericos, si no cabe al ser desplegado, se llenara el campo con ******* para indicar que no cabe. Porcion de un campo alfanumerico puede ser desplegado por medio de subindices, ejemplo:

```
partnum = partnumber[6,10];
```

solo se desplegara los ultimos 5 caracteres del campo. La seccion de atributos es usada para indicar el chequeo de datos, mensaje de errores, valores de default y las siguientes caracteristicas de los campos:

Incluir (Include).

Indica que el dato sera checado en el rango especificado en esta seccion antes de ser aceptado, en caso de que el dato sea erroneo (fuera del rango), entonces se oira un biip y el cursor se posicionara nuevamente en el campo, en espera del dato correcto. El rango puede ser alfanumerico o numerico. Mas de un rango puede ser especificado, ejemplo:

```
f1 = listing_number
     include = (1 to 30000, 32000 to 32500, 30505, 30507);
```

Esto significa que el dato debera estar en el rango 1 a 30000 inclusive o en 32000 a 32500 inclusive o ser igual a 30505 o 30507. Cuando se usa esta opcion, el dato es necesario (no opcional). Cuando no se conoce el dato que se requiere, se pueden introducir blancos (espacios) en campos alfanumericos o cero en campos aritmeticos para que no marque error esta opcion.

Comentarios (comments)

Esta opcion puede ser usada para cambiar el mensaje que es impreso cuando el cursor se mueve a un campo para la entrada de un dato o actualizacion, ejemplo:

```
f1 = listing_number,
     include = (1 to 30000),
     comments = "listing numbers go up to 30,000";
```

Verificacion

Cuando se introducen datos clave , se puede especificar que sean introducidos 2 veces, con la finalidad de evitar al maximo los errores, para lo cual se hace uso de la palabra 'verify', ejemplo:

```
f1 = order_num,
     include = (1 to 49999),
     comments = "order numbers are between 1 and 49999",
     verify;
```

Requerido

Cuando se usa PERFORM, se pueden anadir registros. Cuando el proceso de adicion es iniciado, todos los campos son inicializados con sus valores de default, estos valores son blancos para campos alfanumericos y ceros para aritmeticos, a menos que otro valor de default sea especificado. Si la opcion REQUIRED es incluida en una descripcion de campo, es necesario introducir un dato en este campo, de lo contrario PERFORM indicara que el dato no es opcional.

Busqueda de campos

Cuando el comando LOOKUP es adicionado a la lista de opciones para un campo, permite relacionar archivos para propositos de verificacion. El asterisco (*) precediendo al nombre del campo buscado, indica a PERFORM que el valor que esta siendo introducido ya esta en otro archivo, ejemplo:

```
num = borr_num, lookup from *cust_num;
```

El dato del campo borr_num de un archivo ya debe estar en el campo cust_num de otro archivo. Si la opcion REQUIRED es adicionado para el campo borr_num, los registros no pueden ser a/nadidos sin una entrada valida en este campo, ejemplo:

```
num = borr_num, required, lookup from *cust_num;
```

El atributo

```
num = *cust_num = borr_num, required;
```

enlaza el campo cust_num de un archivo, con borr_num de otro archivo, mientras una entrada es requerida en el campo. Esta entrada debe existir en el archivo de cust_num para ser aceptada.

Default

Para eliminar el default de blancos para campos alfanumericos y ceros para aritmeticos, la opcion DEFAULT puede ser usada, ejemplo:

```
list3 = city, required, default = 'Santa Clara';
```

Noupdate

Cuando se desea que un campo no vaya a ser cambiado (actualizado), se protege por medio de la opcion NOUPDATE, ejemplo:

```
lamt = loan_amt, noupdate;
```

Noentry

No permite inserciones de nuevos registros a la base de datos, solo permite actualizaciones, ejemplo:

```
lamt = loan_amt, noentry;
```

Queryclear

Cuando se efectuan preguntas sobre una pantalla de PERFORM compuesta de dos o mas archivos, PERFORM limpiara la pantalla, dejando solamente los datos de los campos enlazados. Cuando se desea que la pantalla quede completamente limpia, se hace uso de QUERYCLEAR, ejemplo:

```
bnum = cust_num = borr_num, queryclear;
```

Right

Si esta opcion es usada, el contenido del campo sera movido a la derecha en el campo, antes de ser aceptado en la base de datos, ejemplo:

```
lamt = loan_amt, right;
```

Zerofill

Si esta opcion es usada, el campo sera llenado con ceros, en lugar de blancos. El llenado con ceros automaticamente implica justificacion a la derecha, ejemplo:

```
lamt = loan_amt, zerofill;
```

Format

Si se despliega un campo de la base de datos, el cual es de tipo aritmetico, la opcion FORMAT puede ser usada, para indicar cuantas decimales serán impresas cuando el contenido del campo es desplegado, ejemplo:

```
qty = quantity, format = '###.##';
```

el campo quantity de la base de datos, sera desplegado en un campo de 6 posiciones, dejando 2 de ellas para los decimales establecidos.

II.B. Compilacion del archivo creado en editor

Una vez que la especificacion de una pantalla es creada por medio de editor, es compilada. Esta compilacion es realizada usando el comando - - FORMBUILD, ejemplo:

```
formbuild real1
          ^_____ nombre del archivo creado en editor
```

al efectuar la compilacion, se creara un archivo, con ".frm" a/nadido al final del nombre del archivo compilado, en este caso sera "real1.frm". Si la compilacion no es correcta, se generara un archivo, cuyo nombre sera el del archivo compilado (real1 en este caso), con la terminacion ".err", es decir "real1.err", el cual contendra las causas por las que la - - - - compilacion fue erronea.

III. Uso de los comandos QUERY,NEXT y PREVIOUS

Una vez que la compilacion es correcta, PERFORM se puede ejecutar por medio del siguiente comando:

```
perform real1
        ^_____ nombre del archivo compilado
```

con lo que aparecera una pantalla con el siguiente encabezado:

Query Next Previous Add Update Remove File Screen Current Master Detail Output Bye

Lista de bienes raices de las familias residentes en Santa Clara California

E S Q U E L E T O D E F I N I D O A N T E S

Todos los comandos mostrados en la parte superior de la pantalla son ejecutados tecleando el primer caracter (primera letra). Tecleando "a" o "A", se posicionara en la opcion de adicion (Add), una "b" tecleada finalizara la sesion, ya que se posicionara en la opcion "Bye".

Cuando se necesita examinar datos de la base de datos, borrar o - - - actualizar registros, se hace uso primero de la opcion QUERY, tecleando - "q", con lo que el cursor se posicionara en el primer campo del esqueleto definido, esperando la entrada de informacion. Una vez que se introduce el dato, QUERY buscara todos los registros que cumplan con esta condicion (el dato dado), en este momento se esta en condicion de efectuar cambios o simplemente examinar el contenido de los registros que cumplieron con la condicion del dato dado.

Para mas informacion, ver las hojas 28 a 30 del manual de PERFORM.

IV. Cambios en el contenido de una base de datos

IV.A. Actualizacion de datos existentes (comando UPDATE)

El comando UPDATE permite al usuario cambiar el contenido de un registro que esta en la pantalla. Esto se hace tecleando "u", luego hay que posicionarse en el campo que se va a cambiar y para efectuar el cambio o actualizacion una vez que se ha introducido el dato, hay que presionar la tecla "ESC".

IV.B. Adicion de registros (comando ADD)

Cuando una "a" es tecleada, la pantalla es limpiada, excepto para los valores de default y esta lista para recibir datos, el registro es adicionado presionando la tecla "ESC".

IV.C. Eliminacion de registros (comando REMOVE)

El comando REMOVE trabaja de forma similar a los comandos ADD y UPDATE. El registro que sera eliminado, es desplegado en la pantalla con el comando QUERY, NEXT o PREVIOUS y presionando "r" se inicia la eliminacion del registro, en seguida se preguntara que se verifique si el registro que sera borrado es el deseado, para lo cual se debe contestar con una "y" en caso de no haber error en la seleccion del registro, y la eliminacion sera efectuada.

IV.D. Definicion de la salida de los datos (comando OUTPUT)

Haciendo uso de la opcion OUTPUT, se pueden mandar los datos que aparecen en la pantalla a un archivo, para posteriormente ser impreso. Cuando se teclea "o", el sistema preguntara por el nombre del archivo. Se preguntara si el archivo sera creado o si ya existe. Finalmente el usuario puede seleccionar si solo la informacion que esta en la pantalla sera mandada al archivo, o toda la informacion que cumpla con la condicion establecida durante la busqueda.

V. Actividades avanzadas

V.A. Pantallas multiples por formas

Una forma puede ser compuesta de mas de una pantalla, cada pantalla esta definida por un par de llaves { pantalla }.

Cuando PERFORM es ejecutado, aparecera la primera pantalla, y tecleando "s" se pasa a la siguiente pantalla.

V.B. Asociacion de archivos multiples en una forma (comando FILE)

La forma mas simple de desplegar informacion de archivos multiples sobre una pantalla, es dise/nando una forma la cual incluya campos de dos o mas archivos. Para lograr el enlace entre los archivos, en la seccion de atributos se relacionan por medio del signo "=", los nombres de los campos que son comunes en los archivos, ejemplo:

```
attributes
  list1 = agent_name = broker;
```

V.C. Comandos MASTER y DETAIL

Ver hojas 43 a 48 del manual de PERFORM

V.D. Enlaces compuestos

Ver hoja 49 del manual de PERFORM

V.E. Delimitadores de campos

Los campos en las pantallas de PERFORM son encerrados por default en-
parentesis rectangulares '[ ]', esto se puede cambiar especificando los-
delimitadores deseados, en la seccion INSTRUCTIONS de la pantalla de - --
PERFORM, ejemplo:

```
instructions
   delimiters '<>';
```

V.F. Uso de funciones especiales

Ver hoja 51 del manual de PERFORM

VI. Ejemplos

Ver los ejemplos en el manual de PERFORM

TALLER DE DISEÑO DE BASES DE DATOS

INTRODUCCION AL EDITOR VI

ING. DANIEL RIOS ZERTUCHE

SEPTIEMBRE, 1985.

INTRODUCCION AL EDITOR VI

El editor Vi es un editor de pantalla, el cuál está intimamente ligado con el editor "ex" , el cuál es un editor de linea que trabaja bajo la misma filosofía que el "ed" pero más completo. Ya que el Editor "ex" acepta todos los comandos de "ed", no se tratará en este capitulo.

Dentro de las caracteristicas de este editor se encuentran :

- Es un editor que se encuentra normalmente en modo de comandos no en modo de inserción.
- Todos los comandos de "ed" y "ex", pueden efectuarse através de este editor.
- Es adaptable a las necesidades del usuario por medio de la instrucción set.

Tipos de instrucciones para el editor Vi :

- Movimiento del cursor
- Comandos de Inserción
- Comandos de Borrado
- Comandos de Búsqueda y Reemplazo
- Manipulación de Archivos
- Marcado de Bloques
- Comandos Varios
- Sustituciones Globales
- Configuración del Editor

Movimiento del Cursor

Como se mencionó anteriormente el Vi es un editor que entra en modo de comandos. Al encontrarse dentro de este modo en cuando se puede mover através del texto.

Como la mayoría de estos comandos son movimientos sencillos no es necesario dar ejemplos de ellos, nos concretaremos a ejemplificar solo aquellos que creamos necesarios.

Los comandos para moverse a través del texto son los siguientes :

* k : Mueve el cursor hacia arriba, tratando de conservar la misma posición dentro de la columna en la que se encuentra.

* j : Igual que el comando k, pero hacia abajo.

* h : Mueve el cursor hacia la izquierda

* l : Mueve el cursor hacia la derecha

(En algunas terminales funcional también para el movimiento vetical las teclas de "barra-espacio" y "back space").

* ∅ : Mueve el cursor al principio de la linea

* $: Mueve el cursor al final de la linea

* w : Mueve el cursor a la siguiente palabra

* b : Mueve el cursor a la palabra anterior

* e : Mueve el cursor al final de la palabra

*) : Mueve el cursor a la próxima frase (punto)

* (: Mueve el cursor a la frase anterior

* } : Mueve el cursor al próximo párrafo (linea blanca)

* { : Mueve el cursor al párrafo anterior

* W : Mueve el cursor a la próxima palabra delimitada por un espacio.

* B : Mueve el cursor a la palabra anterior delimitada por un blanco

* E : Mueve el cursor al final de la palabra delimitada por un blanco

* H : Mueve el cursor a la esquina superior izq. de la pantalla

* L : Mueve el cursor a la última línea de la pantalla

* M : Mueve el cursor a la línea media de la pantalla

* + : Mueve el cursor al principio de la siguiente línea (la tecla <RETURN> tiene el mismo efecto)

* - : Mueve el cursor al principio de la línea anterior

* nG : Posiciona el cursor en la línea "n" del texto

* ^U : Mueve el cursor dentro de la pantalla doce líneas hacia arriba (scroll up)

* ^D : Mueve el cursor dentro de la pantalla doce línea hacia abajo (scroll down)

* ^F : Mueve el cursor a la próxima página del texto (entendiendo como página el texto que se encuentra en la pantalla)

* ^B : Mueve el cursor a la página previa del texto

* nfx : Mueve el cursor dentro de la línea hasta que encuentra la enésima ocurrencia del carácter "x". Por ejemplo, si tenemos una línea como la siguiente dentro del texto:

esta es una linea dentro del editor VI ú i --- El cursor se encuentra aquí

Después de dar el comando **3fe** por ejemplo, el resultado será :

esta es una linea dentro del editor VI

ú i nueva posición del cursor ---------> []

* **nFx** : Igual que el comando anterior nada más que hacia atrás. es decir que la búsqueda se hará hacia atrás y no hacia enfrente como en el caso anterior.

* **;** : Repite el último comando de "F" ó "f"

* **,** : Repite el último comando, invirtendo la dirección de la búsqueda En este caso, si la última búsqueda fué hacia adelante la hará hacia atrás y viceversa. si tomamos el ejemplo anterior, al dar un ; el cursor se posicionará en :

esta es una linea dentro del editor VI
 Ú
 ¡
--- Aquí después del ;

Comandos de Inserción :

En todos estos comandos salvo el de "r", se entra al modo de inserción, es decir que todo el texto que se teclea a continuación se introducirá en el texto. Para dejar este modo de inserción y volver al modo de comandos se utiliza la tecla <ESC>.

* a : Comienza inserción en la posición que se encuentra a la derecha del cursor.

* i : Comienza inserción en la posición que se encuentra el cursor

* A : Comienza inserción al final de la línea

* I : Comienza inserción al principio de la línea

Ejemplo:

```
i   Posición   del   cursor
v
Esta es otra linea dentro del mismo VI

ô       ôô      ô
i       ii      i
I       ia      A
```

Arriba están las posiciones del cursor después de cada comando dado.

* O : Abre una línea en la posición del cursor y activa modo de inserción.

* o : Abre una línea abajo de la posición del cursor y activa el modo de inserción.

* R : Activa el modo de inserción pero sobreescribiendo el texto actual.

* rx : Reemplaza el caractér sobre el cuál estaba el cursor por el caractér "x"

Comandos de Borrado

* ndw : Borra "n" palabras

* dd : Borra linea 2dd

* ndd : Borra "n" lineas

* ndfx : Borra a partir de la posición del cursor hasta que encuentre la enésima ocurrencia del caractér "x" en la misma linea.

* ndFx : Igual que el anterior nada más que hacia atrás

* x : Borra el caractér sobre el cual se encuentra el cursor

* X : Borra el caractér a la izquierda del cursor

* D : Borra toda la linea y la deja en blanco

En el Modo de Inserción :

* ^H : Borra el último caractér (similar a "back space")

* ^W : Borra la última palabra

Comandos de Búsqueda y Reemplazo

* C : Cambia el resto de la línea

* D : Borra toda la línea

* ncw : Cambia "n" palábras, por el texto de tecleádo hasta dejar el modo de inserción

* ncfx : Sustituye el texto tecleado hasta la enesima ocurrencia del caracter "x" dentro de la misma linea hasta dejar el modo de inserción.

* ncFx : Igual al anterior pero a la inversa.

* s : Cambia el caracter sobre el cual se encuentra el cursor por el texto tecleado.

* S : Cambia la linea sobre la cual se encuentra el cursor sobre el texto tecleado

Manipulación de Archivos

* :w : Graba el texto
* :wq : Graba el texto y salte de vi
* :q : Salida de Vi
* :q! : Salida de Vi, sin guardar los cambios
* :e <nombre> Edita el archivo <nombre>
* :e! Editar el archivo, sin tomar en cuenta los cambios
* :e +<nombre> Edíta el archivo <nombre> y coloca el cursor al final del archivo
* e +n : Edita el archivo a partir de la linea n
* :e # : Editar un archivo alterno
* :w <nombre> Escribie el archivo <nombre>
* :w! <nombre> Reescribe el archivo <nombre>
* :sh Realiza alguna tarea en Shell y regresa a Vi
* :!cmd Correr un comando y regresar a Vi
* :n Editar el siguiente archivo
* n <args> : Especificar nuevos argumentos en la lista
* :f Muestra el archivo (similar a ^G)

Manejo de Bloques

En el caso de que se borren caractéres líneas o palabras, utilizando los comandos de Borrado descritos anteriormente Vi guarda en una memoria temporal el último conjunto de caractéres borrados, los cuales se pueden colocar sobre otra parte del texto por medio del comando "P" que se describe más adelante. Es posible también colocar en la memoria default de Vi cierto texto sin necesidad de borrar el existente o crear memorias etiquetadas en las cuáles se puede guardar una porción de texto.

* **nyfx** : Guarda en la memoria de default los caractéres que se encuentran entre la posición del cursor y la enésima ocurrencia del caractér "x".

* **nyy** : Guarda en la memoria de default las "n" líneas a partir de la línea en la cuál se encuentra el cursor hacia abajo. o **yy**

* **p** : Escribe el texto que se borró o que se guardó en la memoria de default la última véz en la posición siguiente del cursor si es caractér o palabaras o en la línea abajo del cursor en caso de línea o líneas.

* **P** : Escribe el texto que se borró o que se guardó en la memoria de default la última véz en la posición del cursor si es caracter o palabras o en la línea actual (abriendo un espacio) si el línea o líneas

* **n"xdfy** : Borra a partir del cursor hasta la enésima ocurrencia de "y" y lo guarda en la memoria "x".

* **n"xdw** : Borra a partir del cursor "n" palabras y colócalas en la memoria "x"

* **n"xyfz** : Guarda en la memoria "x" todos los caractéres a partir de la posición del cursor hasta la enésima ocurrencia de "z"

* **n"xyFz** : Igual al anterior pero hacia atrás

* **n"xyw** : Guarda en la memoria "x" "n" palabras a partir de la posición del cursor

* **n"xyy** : Guarda en la memoria "x" n-líneas a partir de la posición de cursor hacia abajo

* **"xp** : Coloca en el archivo lo que se tiene en la memoria x

* : n"xdd Borra "n" líneas a partir de la posición del cursor y las coloca en la memoria "x"

Comandos Varios

* ^R : Redibuja la pantalla

* . : Repite el último comando efectuado

* u : Olvida el último comando efectuado sobre la línea

* U : Olvida todos los cambios efectuados sobre la línea

* J : Une la línea actual con la línea abajo del cursor

* tab : En el modo de inserción - mueve el cursor el número de espacios definidos para tabulación

* ^D : En el modo de inserción, se regresa el cursor los "n" espacios definidos en la tabulación hacia atrás

* % : Muestra el paréntesis que corresponde a aquel en que se encuentra el cursor (muy útil para la programación estructurada)

* ^V : Permite la inserción de caractéres especiales como : <ESC> (^[), ^M, etc.

* : ; Permite el uso de un comando de ed (ó ex).

Búsquedas Globales

* /<patrón> : Búsca el patrón a través del texto a partir de la posición del cursor hacia abajo en forma circular (i.e. Si no lo encuentra y llega al final del texto comienza a partir de este hasta llegar nuevamente a la posición del cursor). en cuanto encuentre el primer patrón dentro del texto, posiciona el cursor en ese lugar del texto.

* ?<patrón> : Igual al anterior nada más que en sentido inverso.

* n : Búsqueda a la siguiente ocurrencia del patrón (en sentido circular) hacia abajo.

* N : Búsqueda a la siguiente ocurrencia del patrón hacia arriba.

Sustituciones Globales

Para este tipo de sustituciones se utilizan las capacidades del ex (iguales a las del ed). Para esto se utiliza el comando ":" como por ejemplo :

```
:5,10s/hola/como estas/
```

En este ejemplo sencillo, (es sabido que el potencial de edición de linea puede hacer mucho más que una simple sustitución de dos patrónes), del "ed" o "ex" la primera ocurrencia de "hola" de la linea 5 a la 10 será cambiada por "como estas".

Configuración del Editor

Dentro del editor, hay una serie de variables que van a ayudar a adapter el editor a las necesidades del usuario.

Para ver el valor de esas variables se teclea la instrucción :

```
:set all
```

Al hacer esto aparecerán las variables con su valor default: mencionaremos las más útiles como :

autoindent ai

Esta variable permite que al dar un <cr> permanezca con el mismo márgen de inicio de la linea anterior. La abreviación es : **ai**

noautoindent

Elimina la autoindentación. La abreviación es: **noai**

tabset = <num>

Con esta varible se fija la cantidad de espacios que debe dar cuando se presione la tecla <TAB>. La Abreviación es : ts

shiftwidth <num>

Con esta variable se fija la cantidad de espacios que se regresará el cursor cuando se teclee un ^D en el modo de inserción. La abreviación es : **sw**

redraw

Al estar activada esta variable, el VI redibujará el texto después de un cambio como borrar una linea o insertar caracteres (No tiene abreviación)

noredraw

Desactiva la opción anterior.

list

Al activar esta opción el VI mostrará los caracteres invisibles como <CR> , <^I>, etc.

nolist

Desactiva la opción anterior.

number

Muestra los números de lineas

nonumber

desactiva la opción anterior

Para unilizar o activar estas variables se dá el comando :

:set <conjunto de opciones>

Como por ejemplo, si se quieren las opciones de autoindent, tabset de 3 caracteres y shiftwidth de 5, se dará l- instrucción:

:set ai ts=3 sw=5

Existe también una forma de configurar el VI desde afuera para que se puedan cambiar automáticamente los valores de las variables cada véz que se entre al editor. Esto se hace metiendo la linea de opciones en un archivo que se llame .exrc y que debe encontrarse en el directorio base del usuario. Como por ejemplo, si queremos meter la misma configuración mencionada anteriormente de una forma permanente se podria hacer de una forma sencilla asi :

```
$ cat > .exrc
  set ai ts=3 sw=5
  ^D
$
```

De esta manera cada vez que se invoque el Vi, el editor leerá este archivo y se configurará automáticamente.

Se recomienda tener dentro del archivo **.exrc** las opciones **ai** y **redraw**. Las demás las configurará el usuario de acuerdo a sus necesidades.

DIRECTORIO DE ALUMNOS DEL CURSO "TALLER DE DISEÑO DE BASE DE DATOS" IMPARTIDO EN ESTA DIVISION DEL 6 AL 21 DE SEPTIEMBRE DEL PRESENTE AÑO.

1.- ABURTO GALEANA RAFAEL
S. C.T.

2.- ACOSTA GODINEZ VICTOR
S. A. R. H.
ANALISTA
REFORMA No. 133-6o. PISO
COL. TABACALERA
DELEGACION CUAUHTEMOC
566-89-24

MORELOS 510 EDIF. "C" DEPTO. 304
DELEGACION IZTAPALAPA
560-89-24

3.- BARCENAS EFREN JUAN
S. C.T.

4.- CHAVEZ FALCON GENARO
S. C.T.

5.- DE GANTE MARCOS RAUL
S. C.T.

6.- DURAN ACOSTA ALBERTO
TELEFONOS DE MEXICO

7.- ESCUTIA ACOSTA RAUL

ROSA VENUS No. 25
DELEGACION ALVARO OBREGON
01470 MEXICO, D.F.
680-74-43

8.- FIGUEROA URRINA JAVIER
PROSOFT, S.A. DE C.V.
GERENTE
PUEBLA No. 157-302
COL. ROMA
525-56-65

MADIN No. 53
FUENTES DEL SOL
572-83-17

9.- GARCIA MORALES PABLO ALFONSO
ACEROS FORTUNA, S.A. DE C.V.
ANALISTA PROGRAMADOR
AV. LIC. JUAN FERNANDEZ ALBARRAN No.31
SAN PABLO XALPA
392-50-00

EDIFICIO 4 DEPTO.101 CONJ. SAN BUENAVENTURA
TLALNEPANTLA EDO. DE MEXICO

10.- GARNICA HUITRON GONZALO
S. C. T.

11.- HERNANDEZ DOMINGUEZ JOSE LUIS
S. C.T.
ANALISTA SIST. ESP. DE COMPUTO
EJE CENTRAL LAZARO CARDENAS 567-1er.PISO
COL. NARVARTE
DELEGACION BENITO JUAREZ
519-26-26

2-5-11 FRAC. FCO. VILLA
COL. EXHACIENDA DEL ROSARIO
DELEGACION ATZCAPOTZALCO
02420 MEXICO, D.F.

12.- HERNANDEZ RAMIREZ CARLOS
S. C. T.

13.- IBARRA BLANCAS GUILLERMO
ATISA ATKINS, S.A. DE C.V.
LIDER DE PROYECTO
BAHIA DE CORRIENTES No. 77
COL. ANZURES
DELEGACION MIGUEL HIDALGO
11300 MEXICO, D.F.
250-82-11

VELAZQUEZ DE LEON No. 14-6
DELEGACION CUAUHTEMOC
06470 MEXICO, D.F.
546-69-70

14.- JIMENEZ HERNANDEZ ALEJANDRO
INSTITUTO DE INVESTIGACIONES ELECTRICAS
INVESTIGADOR
DANTE No. 36-4o. PISO
COL. ANZURES
11590 MEXICO, D.F.
511-35-49

ANDADOR AHUEJOTES No. 6
XOCHIMILCO
511-32-26

15.- LANDEROS AVILES CARLOS F.
COLEGIO SUP. DE AGRICULATURA TROPICAL
INVESTIGADOR
CARDENAS TAB.

ZACATECAS No. 178-3
COL. ROMA
DELEGACION CUAUHTEMOC
564-83-23

16.- LOPEZ LOPEZ ROBERTO
TELEFONOS DE MEXICO

17.- LOPEZ LOPEZ HUMBERTO
ACEROS FORTUNA, S.A. DE C.V.
ANALISTA PROGRMADOR
LIC. JUAN FDZ. ALBARRAN No. 32
COL. SAN PABLO XALPA
TLALNEPANTLA, EDO. DE MEXICO
392-50-00

TENORIOS 222 EDIF. 26 DEPTO. 3
EXHACIENDA DE COAPA
392-50-00

18.- LOPEZ MENDIZABAL VICTOR
S. A. R. H.
ANALISTA
AV. INSURGENTES SUR No. 30
COL. JUAREZ
DELEGACION CUAUHTEMOC
591-18-35

ING. JOSE J. REYNOSOS No. 68
COL. CONSTITUCION DE 1917
DELEGACION IZTAPALAPA
09260 MEXICO, D.F.
691-37-79

19.- LUNA LOMELI JUAN JOSE
S. C.T.

20.- MAGAÑA CARRILLO JUAN F.
S. C.T.
ANALISTA DE SIST. EN VIAS TERRESTRES
AV. UNIVERSIDAD Y XOLA
COL. NARVARTE
DELEGACION BENITO JUAREZ
519-73-60 y 519-07-30

AV. UNIVERSIDAD No. 1810-A-1
DELEGACION COYOACAN
04310 MEXICO, D.F.

21.- MARTINEZ CAMACHO JOSE LUIS
PETROLEOS MEXICANOS
JEFE DE ANALISTAS
MARINA NACIONLA No. 329
COL. ANAHUAC DELEG. CUAUHTEMOC
250-53-56

SERENATA No. 35 P.B.
VILLAS DE LA HACIENDA
ATIZAPAN, EDO. DE MEXICO
250-53-56

22.- MONTERRUBIO ESCOBAR MANUEL DELFINO
PEMEX
ANALISTA PRINCIPAL
MARINA NACIONAL No. 329
COL. ANAHUAC
250-26-11 ext. 21902

RETORNO No. 36 DE CECILIO ROBELO
40-B-6
DELEGACION VENUSTIANO CARRANZA
15900 MEXICO, D.F.
250-26-11 ext. 21902

23.- MUÑIZ GOMEZ JORGE
S. C. T.
ANALISTA
EJE CENTRAL LAZARO CARDENAS No. 567
530-30-60 ext. 199 y 685

AV. OCEANIA No. 54 DEPTO. 8
COL. ROMERO RUBI
DELEGACION VENUSTIANO CARRANZA
15400 MEXICO, D.F.

24.- PICAZO SALINAS JORGE
DIREC. GRAL. CARRET. FEDERALES S.C.T.
ANALISTA DE SISTEMAS
CENTRO SCOP BASAMENTO
AV. UNIVERSIDAD Y XOLA
COL. NARVARTE
DELEGACION BENITO JUAREZ
03022 MEXICO, D.F.
530-02-24

BARRANCA DE PILARES No. 1
COL. LAS AGUILAS
DELEGACION ALVARO OBREGON
01710 MEXICO, D.F.

25.- RAMIREZ MAGAÑA JOSE ROBERTO
S. C. T. D.G.O. M.
JEFE OFNA. INVEST. OPERAD.
PROVIDENCIA No. 807-4o. PISO
COL. DEL VALLE
DELEGACION BENITO JUAREZ
03100 MEXICO, D.F.

26.- RODRIGUEZ LUGO SERGIO DANIEL
DIREC. GRAL. CARRET. FEDERALES
ANALTSTA PROGRAMADOR
XOLA Y AV. UNIVERSIDAD
COL. NARVARTE
DELEGACION BENITO JUAREZ
03028 MEXICO, D.F.
530-02-29

CJON. BOJOQUEPA No. 30-30
LA SUNCION
XOCHIMILCO 16040
676-70-26

27.- ROMAHN DIAZ JUAN
PETROLEOS MEXICANOS
JEFE DE DEPARTAMENTO
MARINA NACIONAL

DELFINOS No. 5 F. DE SATELITE
ATIZAPAN DE ZARAGOZA
572-69-89

28.- SAMPERIO SANCHEZ MERRY
S. C.T.
ANALISTA ESP.
XOLA Y AV. UNIVERSIDAD
DELEGACION BENITO JUAREZ
530-11-98

CALLE 647 No. 237
COL. UNIDAD SAN JUAN DE ARAGON
DELEGACION GUSTAVO A. MADERO
07420 MEXICO, D.F.
796-33-84

29.- SANCHEZ VILLARREAL FRANCISCO
S. C.T.

30.- SERAFIN CADELARIO ILDEFONSO
DIREC. GRAL. ING. DE SISTEMAS
ANALISTA PROGRAMADOR
AV. MICHOACAN S/N
COL. TEPALCATES
DELEGACION IZTAPALAPA
692-00-77

CANAL DE GARAY No. 97
LOS ANGELES
DELEGACION IZTAPALAPA
09710 MEXICO, D.F.

31.- TREJO FLORES LIDIA
DIREC. GRAL. ING. DE SISTEMAS
ANALISTA ESP.
DIREC. GRAL. ING. DE SISTEMAS
COL. TEPALCATES
DELEGACION IZTACALA
09280 MEXICO, D.F.
691-70-56

GONZALEZ CAMARENA No. 9
COL. JACARANDA
DELEGACION IZTAPALAPA
09280 MEXICO, D.F.
691-77-64

32.- VELAZQUEZ QUEZADA JORGE
S. C.T.
ING. ESP. EN TELECOMUNICACIONES
EJE CENTRAL LAZARO CARDENAS No. 267
COL. NARVARTE
03028 MEXICO, D.F.
530-20-99

GLORIETA DE BUCARELI No. 39
07000 MEXICO, D.F.
797-54-72

33.- VIZCAINO SAHAGUN CARLOS E.
INSTITUTO DE INVESTIGACIONES ELEC.
INVESTIGADOR
DANTE No. 36-4o. PISO
COL. ANZURES
11590 MEXICO, D.F.
511-35-44

COPILCO 76, 82-301
COL. COPILCO EL BAJO
DELEGACION COYOACAN
04340 MEXICO, D.F.
548-26-35