19
20

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Escuela Nacional de Estudios Profesionales

" ARAGON "

LENGUAJE SQL APLICADO A BASE DE DATOS RELACIONALES

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A:

CESAR JIMENEZ MEJIA

TESIS CON
FALLA DE ORIGEN

ENEP

ARAGON

San Juan de Aragón, Edo. de Méx., 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DERECHOS RESERVADOS © PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

UNIVERSIDAD NACIONAL AVENIDA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ARAGÓN
DIRECCION

CESAR JIMENEZ MEJIA
P R E S E N T E .

En contestación a su solicitud de fecha 31 de mayo del - año en curso, relativa a la autorización que se le debe- conceder para que el señor profesor, Ing. LUIS LORENZO - JIMENEZ GARCIA, pueda dirigirle el trabajo de Tesis deno minado "LENGUAJE SQL APLICADO A BASE DE DATOS RELACIONA- LES", con fundamento en el punto 6 y siguientes del Re- glamento para Exámenes Profesionales en esta Escuela, y toda vez que la documentación presentada por usted reune los requisitos que establece el precitado Reglamento; me permito comunicarle que ha sido aprobada su solicitud.

Aprovecho la ocasión para reiterarle mi distinguida con - sideración.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón, Edo. de Méx., Junio 28, 1993.
EL DIRECTOR

M. en I. CLAUDIO C. MERRIFIELD CASTRO

C.c.p. Lic. Alberto Ibarra Rosas.- Jefe de la Unidad Académica.
- Ing. Juan Gastaldi Pérez.- Jefe de Carrera de Ingeniería en Computación.
- Ing. Luis Lorenzo Jiménez García.- Asesor de Tesis.

CCMC'AIR'eom.

Un ingeniero oraba así

¡Señor, ayudame a construir un mundo en donde reine la paz y la amistad entre los pueblos!. De esta manera se cumplirán fielmente tus palabras:" Amense los unos a los otros ".

¡Señor, ayudame a edificar una sociedad en la cual la injusticia sea desterrada para siempre, la corrupción encarcelada eternamente y el egoísmo aniquilado por la fuerza del amor!.

¡Señor, ayudame a formar una familia en la que el vínculo indestructible le sea el mutuo afecto, basado en la plena confianza y caridad solicita!.

¡Señor , ayudame a levantarme del fango del pecado y poder así, elevarme hasta las alturas celestiales de la humildad, la bondad, la fe , la caridad y la esperanza!.

¡Señor tu que eres el creador de universo, has que sea posible todo lo anterior!.

DEDICATORIAS

A Dios por darme esta oportunidad en mi vida de terminar una profesión.

A mis padres Jesús y Ma. de la Luz por su valioso esfuerzo y sacrificios durante la terminación de mis estudios.

A mi esposa Cynthia por todo el cariño y comprensión durante la terminación de este trabajo.

A mis hermanos David y Marcelo como un estímulo de superación.

A toda mi familia en general.

AGRADECIMIENTOS ESPECIALES

A la Lic. Conchita por toda su valiosa ayuda y motivación durante la elaboración de este trabajo.

Al Ing. Luis L. Jiménez García por toda su gran ayuda durante la elaboración de este trabajo.

A la Universidad Nacional Autonoma de México, en especial a la ENEP ARAGON por darme la oportunidad de haber estudiado ahí.

INDICE

INTRODUCCION

En la actualidad, las empresas están dependiendo cada vez más de la disponibilidad y la precisión de la información. Por lo que las bases de datos relacionales han proporcionado soluciones a muchos de los problemas de informática y de los usuarios finales.

En el presente trabajo se expone al lenguaje estructurado de consulta (SQL) como el lenguaje manejador de la base de datos relacional de la compañía ORACLE, cuyo objetivo es mostrar el uso y aplicación del lenguaje SQL.

El CAPITULO I comprende algunos conceptos básicos de bases de datos, los diferentes tipos o modelos de bases de datos que existen y ciertas definiciones como Sistema Manejador de Base de Datos, el Administrador de la base de datos, base de datos distribuida; todo lo anterior con la finalidad de compreder mejor los siguientes capítulos.

El CAPITULO II comprende los conceptos de base de datos relacionales , los operadores relacionales, el modelado de un problema utilizando el modelo de Entidad-Relación y una breve historia de la compañía de base de datos relacionales ORACLE.

El CAPITULO III comprende una introducción breve del lenguaje estructurado de consulta (SQL) y además de mostrar los comandos de SQL con ejemplos que permitirán un aprendizaje más óptimo de dicho lenguaje.

El CAPITULO IV comprende todos los comandos adicionales de SQL llamados comandos de SQL*Plus que nos permitirán obtener reportes más sofisticados que si los hicieramos con comandos de SQL y por último aplicar dichos comandos a un Sistema de Comprobación Presupuestal.

CAPITULO

BASE DE DATOS

1

I.1 CONCEPTOS BASICOS

A continuación se definen los términos para definir las bases de datos.

Byte

Un byte es la parte más pequeña de información en una base de datos y esta compuesta de ocho bits.

Campo

Un campo identifica una posición en un registro y puede estar formado por cualquier número de bytes.

Registro

Un registro es un grupo de campos afines de información considerados como una unidad.

Archivo

Un archivo es un conjunto de registros de formato idéntico y cada uno contiene una serie de campos.

Base de datos

Una base de datos se define como un conjunto de datos relacionados entre sí, almacenados, estructurados, no redundante; cuya finalidad es servir a una o más aplicaciones; los datos son independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y modificar o extraer los datos almacenados.

Enfoque de la base de datos

El enfoque de la base de datos nos permite:

- Controlar la redundancia.
- Mantener la consistencia.
- Lograr la integración de los datos.
- Compartir los datos entre las diferentes aplicaciones.
- Cumplir con los estándares.
- Tener facilidad en el desarrollo de aplicaciones.
- Independencia entre los datos y programas.
- Reducir el mantenimiento a los programas.

Diccionario de datos

Un diccionario de datos es aquel que guarda información acerca de la base de datos. También se puede definirse como una herramienta para identificar y clasificar los datos almacenados en la base de datos.

I.2 EL SISTEMA MANEJADOR DE BASE DE DATOS

Un Sistema Manejador de Base de Datos(DBMS, Data Base Management System) es un conjunto de datos relacionados entre sí y un conjunto de programas para tener acceso a esos datos. Al conjunto de datos se le conoce como base de datos y su contenido de información esta organizada de una manera determinada. La función primordial de un DBMS es crear el ambiente en que sea posible guardar y recuperar información de la base de datos en forma eficiente y conveniente.

Características:

- Representación de datos a través de tablas.
- Desarrollo de aplicaciones a través de herramientas de alta productividad.
- Fexibilidad en el manejo de las estructuras y de los datos en el tipo de consultas.
- Diccionario de datos integrado
- Soporte a todos los operadores relacionales.

Algunos de los objetivos de un DBMS son:

- Minimizar la consistencia de los datos, es decir, no tener datos repetidos y no almacenar datos derivados.
- Garantizar la consistencia de los datos, es decir, obtener la misma información por peticiones similares en un momento dado.
- Integridad de los datos, es decir, que cumpla con las reglas dictadas por políticas o normas de la empresa.
- Seguridad en los datos, es decir , que los datos tengan protección contra accesos, modificaciones o pérdidas ya sea en forma intencional o no intencional.
- Controlar la concurrencia, es decir, que varios usuarios puedan accesar a la misma información al mismo tiempo sin tener problemas en los datos.
- Proteger los datos contra fallas, es decir, que tenga la capacidad de restaurar la integridad y consistencia después de una falla del sistema.
- Tener un diccionario de datos, que es la capacidad que da al manejador de la base de datos de poder tener la descripción de los los datos que estan almacenados en la base de datos.

Ventajas:

- Fácil de usar.
- Fácil obtener respuestas.
- Fácil insertar y actualizar datos.
- Fácil cambiar la estructura de los datos.
- La navegación es responsabilidad del DBMS, no del programador.
- Todas las consultas son posibles.

Generaciones de los DBMS's

Existen 3 generaciones de los DBMS's que son las siguientes:

1)Jerárquico (1960's):

Un programador únicamente puede especificar relaciones como un padre y múltiples hijos.

2)Red (1970's):

Un programador puede especificar relaciones múltiples padres y múltiples hijos.

3)Relacional (1980's):

El sistema establece automáticamente todas las relaciones posibles. Por ejemplo, ORACLE V.2 fue el primer RDBMS (Relational Data Base Management System) comercial.

I.2 El ADMINISTRADOR DE LA BASE DE DATOS

El Administrador de la base de datos (DBA, Database Administrator) es un programa que organiza la información, este puede ser una persona o grupo responsable que se encargue de supervisar el buen funcionamiento del software y hardware de la base de datos y además introducir las necesidades de los usuarios en la base de datos.

Al DBA le interesa incluir lo siguiente:

- Un software de instalación y mantenimiento.
- Afinar la base de datos par un óptimo uso.
- Seguridad en los datos.
- Almacenamiento de datos.
- Disponibilidad de datos.

- Recuperación de datos.

I.4 MODELOS DE BASES DE DATOS

Existen varios modelos alternativos para visualizar y manejar datos a un nivel lógico independiente de cualquier estructura física en que se basen.

Los modelos de bases de datos que existen son:

- Modelo jerárquico.
- Modelo de red.
- Modelo relacional.

Modelo jerárquico

En el tipo de base de datos jerárquica su estructura lógica en la cual se sostiene la base de datos es un árbol. Un árbol se compone de un nodo raíz y varios nodos sucesores ordenados jerárquicamente.

El nodo colocado en la parte superior es llamado padre y los nodos inferiores son los hijos (ver la fig. I.4.1).

En el sistema jerárquico las conexiones entre archivos no dependen de la información contenida en ellos, se definen al inicio y son fijos.

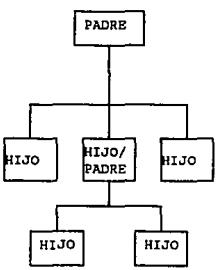


Fig. I.4.1 Estructura del modelo jerárquico

Modelo de red

En el tipo de base de datos de red los datos se representan como registros ligados formando un conjunto de datos intersectados. La base de datos de red, a diferencia de las jerarquicas, permite cualquier conexión entre entidades, es decir, se puede representar relaciones de muchos a muchos. En una red, un hijo puede tener varios padres y varios hijos a la vez (ver la figura I.4.2)

Características:

- Representación de los datos similar al modelo jerarquico.
- Se puede tener relaciones de un hijo a varios padres.
- En una red, un hijo puede tener varios hijos y varios padres a la vez.
- Reduce el tiempo necesario para encontrar la información.

Desventajas:

- Resulta difícil definir nuevas relaciones.
- Es complicado darle mantenimiento ya que cualquier cambio en la estructura requiere una descarga den los datos.

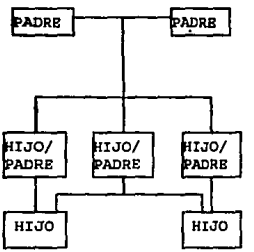


Fig. I.4.2 Estructura del modelo de red

Modelo relacional

La estructura lógica de una base de datos relacional está basada en la representación de entidades mediante tablas, las cuales constan de columnas y renglones. Las relaciones entre tablas se llevan a cabo a través de conjuntos de columnas que se tengan en común (ver la figura I.4.3).

	COLUMNA1	COLUMNA2
RENGLON 1		
RENGLON 2		
RENGLON n		

Fig. I.4.3 Estructura del modelo relacional

La ventaja de los sistemas relacionales es poder modificar la información sin la preocupación de especificar las combinaciones entre registros.

Características:

- La información esta organizada en forma de tablas.
- Las categorías de información están listadas a lo largo de la parte superior de cada tabla.
- Los casos individuales están listados hacia abajo.
- En ésta forma se puede visualizar, entender y utilizar la información de una forma sencilla.
- Se tiene fexibilidad suprimiendo la jerarquía entre campos.

I.5 BASES DE DATOS DISTRIBUIDAS

Las bases de datos distribuidas es una colección de bases de datos almacenadas en más de un CPU (Unidad Central de Proceso), dentro de la cual el usuario puede recibir la información como si fuera una simple base de datos. Cuando en realidad son varias bases de datos pequeñas.

Características:

- Cada máquina de la red posee capacidad de procesamiento autónomo y puede efectuar aplicaciones locales.
- Cada máquina participa también en ejecución de cuando menos una aplicación que requiere accesar a datos de varias máquinas por medio de un subsistema de comunicaciones.

Proceso distribuido

El proceso distribuido ocurre cuando una aplicación en el CPU accesa a una base de datos en otro CPU.

Algunos de los beneficios del proceso distribuido son los siguientes:

- Puedes usar la mayoría del hardware apropiado para tu trabajo.
  Por ejemplo, las computadoras personales (PC's) pueden ser usadas para rápidos procesamientos, mientras las computadoras mini o mainframe con rápida Entrada/Salida y economía pueden utilizarse para almacenar el dato centralmente y accesar los datos por medio de las PC's.
- Puedes soportar más usuarios con un hardware económico. Más procesos pueden ocurrir en tu PC y la capacidad puede incrementarse para comprar PC's mejores que hardware para la mainframe.

CAPITULO

BASE DE DATOS RELACIONALES

II

II.1 CONCEPTOS DE BASES DE BASES DE DATOS RELACIONALES

Tablas, renglones y columnas

Una tabla es una estructura que guarda datos en la base de datos relacional. Y una tabla esta compuesta de renglones y columnas. En donde cada columna contiene un tipo de información y cada renglón está compuesto de varias columnas que contienen cada una un valor.

Ejemplo:

Si la columna de salario (SAL) en el renglón de Jorge tiene un valor de 2975.

	NUM_EMP	NO_EMP	PUESTO	FEC_ING	SAL	COM	NUM_DPTO
R	7329	PEDRO	ANALISTA	17-JUN-92	1600	200	20
E	7499	ALEJANDRO	OPERADOR	20-FEB-92	800		20
N	7521	ALBERTO	CAPTURISTA	24-OCT-91	1250		20
G	7566	JORGE	JEFE	02-APR-90	2975	200	20
L							
O							
N							
E							
S			C O L U M N A S				

Tablas relacionadas

La información de una tabla puede estar relacionada con la información de otra tabla.

Por ejemplo, cada empleado de la tabla EMP tiene un número de departamento (NUM_DEPTO) que se refiere a un número de departamento en la tabla DEPTO.

NUM_DPTO	NO_DPTO	LOC
40	FINANZAS	D.F
20	INFORMATICA	EDO.MEX.

Algunas ventajas de usar tablas son las siguientes:

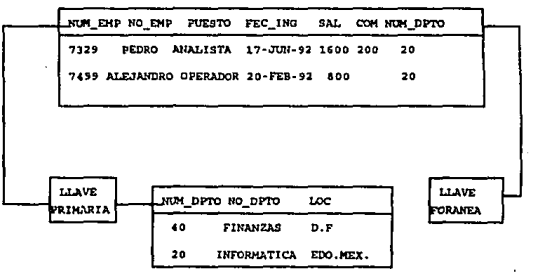
- En el formato columna/renglón de las tablas se pueden visualizar fácilmente los datos.
- Se puede aplicar la teoría de conjuntos y el algebra relacional en las tablas.
- La información de una tabla se puede relacionar con otra, esto permite organizar la información en unidades independientes y fáciles de manejar.

- Se puede relacionar información de 2 o más tablas.

Indices, llaves y llaves foráneas

Los índices son estructuras almacenadas en la base de datos que el usuario las crea para hacer más rapidas las consultas. Las llaves son un concepto lógico. Las llave primaria es una columna (o combinación de columnas) que pueden ser usadas únicamente identificando algún renglón en una tabla. La llave foránea es una columna o (o grupo de columnas) en una tabla que corresponde a una llave primaria en otra tabla. Las llaves foráneas son usadas cuando combinan datos de múltiples tablas (un Join de tablas).

Ejemplo:



II.2 OPERADORES RELACIONALES.

El interés en el enfoque relacional se debe en gran parte al trabajo realizado del Dr. E. F. Codd, quien en Junio de 1970 publicó un artículo "A Relational Model of Data for large Shared Data Bank" Modelo Relacional de Datos para grandes proporciones de Bancos de Datos.

Los operadores relacionales son:

Unión

El operador unión acepta como entrada 2 tablas con las mismas columnas en el mismo orden, y produce como resultado todas las columnas y renglones de ambas tablas. Si existe algún

renglón con la misma información en ambas tablas, en la tabla que se genera al aplicar el operador unión ese renglón sólo aparece una vez.

Ejemplo:

Tenemos las siguientes tablas:

TABLA: EMPLEADOS ANTIGUOS TABLA: EMPLEADOS NUEVOS

NUM_EMP	NO_EMP	SAL
1	PEDRO	12000
2	LUIS	
3	RAUL	36000

NUM_EMP	NO_EMP	SAL
3	RAUL	36000
4	LORENA	24000
5	IVON	24000

El resultado de la unión es:

NUM_EMP	NO_EMP	SAL
1	PEDRO	1200
2	LUIS	
3	RAUL	36000
4	LORENA	24000
5	IVON	24000

Intersección

El operador de intersección selecciona de ambas tablas los renglones que tengan exactamente la misma intersección en todas las columnas.

Ejemplo:

Tenemos las siguientes tablas

TABLA: EMPLEADOS ANTIGUOS TABLA: EMPLEADOS NUEVOS

NUM_EMP	NO_EMP	SAL
1	PEDRO	12000
2	LUIS	
3	RAUL	36000

NUM_EMP	NO_EMP	SAL
3	RAUL	36000
4	LORENA	24000
5	IVON	24000

El resultado de la intersección es:

NUM_EMP	NO_EMP	SAL
3	RAUL	36000

Diferencia

El operador diferencia acepta como entrada 2 tablas que contengan al menos una columna en común, en donde la tabla resultante tendrá todas las columnas de la primer tabla y los renglones que no aparezcan en la segunda tabla.

Ejemplo:

Tenemos las siguientes tablas:

TABLA:DEPTO

COD_DPTO	NO_DPTO
VE	VENTAS
NO	NOMINA
IN	INVEST.
ME	MERCAD.
RE	RESULT.

TABLA: EMP

NUM_DPTO	NO_EMP	SAL	COD_DPTO
1	PEDRO	12000	VE
2	LUIS		NO
3	RAUL	36000	
4	LORENA	24000	
5	IVON	24000	NO

El resultado de la diferencia es:

COD_DPTO	NO_DPTO
IN	INVEST.
ME	MERCAD.
RE	RESULT.

Proyección

El operador de proyección tiene como entrada una tabla y produce como resultado sólo aquellas columnas especificadas por el usuario. El orden en el cual aparecen las columnas es el que se indica cuando se hace la proyección. Y el número de columnas que se pueden proyectar es como máximo el mismo número de columnas de la tabla y como mínimo una sola columna.

Ejemplo:

Tenemos la siguiente

TABLA: EMP

NUM_DPTO	NO_DPTO	SAL	COD_DPTO
1	PEDRO	12000	VE
2	LUIS		NO
3	RAUL	36000	
4	LORENA	24000	
5	IVON	24000	NO

La proyección de las columnas NO_EMP y NUM_EMP es:

NO_EMP	NUM_EMP
PEDRO	1
LUIS	2
RAUL	3
LORENA	4
IVON	5

Selección

El operador de selección acepta una sola tabla como entrada y produce como resultado las mismas columnas que contiene la tabla de entrada y los renglones que sean especificados por el usuario. Las condiciones de selección de renglones pueden ser de varios grados de complejidad y pueden incluir a los operadores booleanos (AND, OR y NOT), pudiendose utilizar paréntesis para indicar la precedencia de operación. Las comparaciones pueden realizarse con valores literales, valores contenidos en las columnas o expresiones matemáticas que involucren valores literales de las columnas.

Ejemplo:

Tenemos la siguiente tabla

TABLA: EMP

NUM_DPTO	NO_DPTO	SAL	COD_DPTO
1	PEDRO	12000	VE
2	LUIS		NO
3	RAUL	36000	
4	LORENA	24000	
5	IVON	24000	NO

La selección de todas los empleados que trabajan en el departamento de finanzas son:

NUM_EMP	NO_EMP	SAL	COD_DPTO
2	LUIS		NO
5	IVON	24000	NO

Join

El operador de Join acepta como entrada 2 o más tablas, teniendo cada una al menos una columna en común con las otras tablas y produce como resultado todas las columnas de la tabla de entrada y los renglones que se concatenan con aquellos renglones cuyos valores en las tablas de entrada cumplan la condición que indica el usuario para hacer el Join. Los operadores relacionales para indicar las condiciones de Join pueden ser >, <, =, != . Las columnas en común solo se muestran una vez.

Ejemplo:

Tenemos las siguientes tablas

TABLA: DEPTO

COD_DPTO	NO_DPTO
VE	VENTAS
NO	NOMINA
IN	INVEST.
ME	MERCAD.
RE	RESULT.

TABLA: EMP

NUM_DPTO	NO_EMP	SAL	COD_DPTO
1	PEDRO	12000	VE
2	LUIS		NO
3	RAUL	36000	
4	LORENA	24000	
5	IVON	24000	NO

El Join con la columna en común del COD_DPTO es:

COD_DPTO	NO_DPTO	NUM_DPTO	NO_EMP	SAL
VE	VENTAS	1	PEDRO	12000
NO	NOMINA	2	LUIS	
NO	NOMINA	5	IVON	24000

II.3 MODELO DE ENTIDAD-RELACION

El modelo de Entidad-Relación es una técnica para definir las necesidades de información de cualquier empresa. Esta técnica involucra conceptos que se identifican con varios objetos de importancia para la empresa, a los cuales se les denomina "Entidades" , a las características de

dichos objetos se les denomina "Atributos" y a como se relacionan estos objetos entre sí se les denomina "Relaciones".

Todos estos conceptos se modelan a través de cierto tipo de esquemas gráficos, los cuales muestran a los usuarios una manera más sencilla y práctica de visualizar sus necesidades de información.

Entidad

Una entidad es una persona, cosa o lugar, que cae dentro del alcance del sistema, acerca de la cual el sistema debe mantener, correlacionar y desplegar información.

La entidad se representa por medio de una caja con las esquinas redondeadas y dentro de ésta se escribe el nombre de la entidad, el cual debe estar en singular, y cada entidad debe tener un nombre único dentro del sistema. El esquema de una entidad se muestra en la figura II.3.1

Fig. II.3.1 Esquema de una entidad

La entidad dentro del modelo relacional se representa por medio de una tabla, donde el nombre de la tabla corresponde al nombre de la entidad. Además, cada tabla debe contener una columna que identifique de forma única a cada renglón de ésta. Esta columna recibe el nombre de "llave primaria (PK)", la cual no debe contener valores nulos ni duplicados.

Ejemplo:

Atributo

Un atributo es una característica o cualidad de una entidad o relación, que cae dentro del alcance del sistema, acerca del cual el sistema debe mantener , correlacionar y desplegar información.

Para representar uno o varios atributos se describe el nombre del atributo dentro de la entidad, opcionalmente se puede mencionar ejemplos.

El esquema de un atributo se muestra en la figura II.3.2

Fig. II.3.2 Esquema de un atributo

Los atributos dentro del modelo se representan por medio de columnas dentro de una tabla.

Ejemplo:

PRODUCTO

COD_PROD	DESCRIP
PK	
100860	DISCOS
100861	CINTAS
100862	IMPRESORAS
100863	PAQUETES

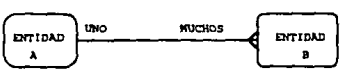
Relación

Una relación es la unión que existe entre dos o más entidades, la cual debe caer dentro del alcance del sistema, acerca de la cual el sistema debe mantener, correlacionar y desplegar información.

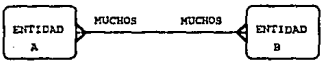
Entre las entidades pueden existir relaciones o asociaciones, que pueden ser de los siguientes tipo:

1) Relaciones de uno a uno.

2) Relaciones de uno a muchos.



3) Relaciones muchos a muchos.



Las relaciones dentro de un modelo relacional se representan de la siguiente forma:

- Relación uno a uno entre 2 entidades se modela dibujando la llave primaria (PK) de la tabla con más renglones como llave foránea (FK) en la otra tabla. Una llave foránea es una columna o más columnas que son llave primaria en otra tabla y una llave foránea permite nulos y valores duplicados.

Ejemplo:

PRODUCTO

NUM_PROD
PK
100860
100861
100862
100863

PRECIO

CVE_PREC	NUM_PROD
PK	PK
1	100860
2	100861
3	100862
4	100863

- Relaciones uno a muchos o muchos a uno entre 2 entidades se modela dibujando la llave primaria (PK) de la tabla que tiene la correspondencia de uno como llave foránea (FK) en la otra tabla.

Ejemplo:

CLIENTE

NUM_CLIE
PK
100
101
102
103

ORDEN

NUM_ORD	NUM_CLIE
PK	FK
600	101
601	101
602	102
603	...

- Relación muchos a muchos entre 2 entidades se modela dibujando una tercer tabla, la cual se compone de una llave primaria (PK) compuesta de 2 columnas, las cuales son llaves primarias y foráneas a la vez.

Ejemplo:

ORDEN

NUM_ORD
PK
600
601
602
603

PRODUCTO

COD_PROD
PK
100860
100861
100862
100863

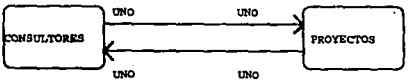
ORDEN/PRODUCTO

NUM_ORD	COD_PROD
PK	
FK	FK
600	100860
601	100861
602	100862
603	100863

A continuación se muetran 2 ejemplos prácticos utilizando el modelo de Entidad-Relación.

1. El usuario es gerente de una firma de consultoría. Cada proyecto (identificado por un número único de proyecto) es la responsabilidad de un consultor. Cada consultor es responsable de sólo un proyecto, ya que se dedica tiempo completo a él. Los consultores son identificados por un número de consulta. Cada cliente sólo tiene un proyecto atendido por la compañía. Dado la gran demanda de la compañía se tiene proyectos pendientes por falta de personal.

Una solución es:



PROYECTOS

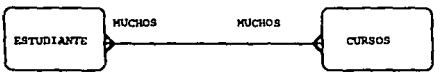
NUM_PROY
PK
204
205
206
207

CONSULTOTES/PROYECTOS

NUM_CON	NUM_PRO
PK	FK
1	
2	207
3	206
4	205

2. El usuario es responsable del registro de estudiantes de un colegio privado. A cada estudiante se le asigna un número único de estudiante y puede inscribirse en uno o más cursos. Los cursos tienen códigos únicos y a ellos se inscriben en promedio de 20 a 30 estudiantes.

Una solución es:



ESTUDIANTES

NUM_EST
PK
1
2
3

CURSOS

COD_CUR
PK
ING-1
FRAN-1
ALEM-1

ESTUDIANTES/CURSOS

NUM_EST	COD_EST
PK	
FK	FK
1	ING-1
2	FRAN-1
3	ALEM-1
2	ING-1

II.4 ORACLE

Historia

La compañía empieza en 1977. En 1979 Larry Ellison y Bob Miner (dos de las personas más importantes dentro de la compañía actualmente) implementarón la idea del reporte técnico de IBM, el cual hablaba sobre un sistema de recursos humanos del cual surgió el lenguaje SQL (en inglés Structured Query Language, Lenguaje Estructurado de Consulta). Ellos escribierón el primer SQL basado en una base de datos relacional. Otra importante decisión fue tomada en 1979, escribieron el código en lenguaje C. Ellos tomaron la idea de IBM y la hicierón portable a través del lenguaje SQL, de hecho la máquina que probó dicho desarrollo fue una computadora DEC llamada PDP11. ORACLE (de ORACULO) toma su nombre de un primer contrato, un proyecto confidencial para el gobierno, el password de ese proyecto de seguridad era precisamente ORACLE. Larry y Bob decidierón llamar a la compañía como dicho concurso. La compañía inicialmente se encontaba situada en Melo Park. Actualmente se encuentra en San Francisco.

Hasta ahora la compañía se encuentra en 95 ciudades alrededor del mundo. La compañia ORACLE ocupa el primer lugar de las compañias de bases de datos relacionales comercialmente y tiene continuamente las primeras inovaciones en el campo de los sistemas manejadores de bases de datos relacionales (RDBMS) . La compañía ORACLE ofrece un RDBMS portable, compatible y herramientas poderosas para usuarios.

Arquitectura

La arquitectura cliente-servidor explota al máximo la tecnología más avanzada, incluyendo aquellos sistemas que incluyen multiprocesadores (SMPs). La base de datos de ORACLE permite accesar en una sola consulta, datos que se encuentam en servidores remotos.

Algunas de las características de la compañia ORACLE son:

- Compatibilidad: es hablar de una gama de productos, los cuales se rigen bajo los estándares de la industria de las bases de datos relacionales, lo que conduce que una base de datos ORACLE sea compatible con otras bases de datos en el mercado.

- Portabilidad: una aplicación de ORACLE puede ser transportada de una máquina a otra, con diferentes sistemas operativos y esto no provoca ningún cambio o modificación a dicha aplicación.

- Conectividad: la arquitectura de ORACLE permite que los datos y las aplicaciones residan en diferentes computadoras, plataformas, sistemas operativos y ambientes de red, la idea es

tener una conectividad completa.

- **Capacidad:** tener la capacidad de manejo de grandes volumenes de información facilita el uso de aplicaciones grandes.

- **Productividad:** ORACLE cuenta con diversas herramientas y productos que apoyan en la forma de tomar decisiones

CAPITULO

LENGUAJE ESTRUCTURADO DE CONSULTA (SQL)

III

III.1 INTRODUCCION A SQL

SQL se introdujo como lenguaje de consulta del sistema R, el cual fue un proyecto de investigación que se desarrollo en 1974 por IBM. El objetivo del proyecto era demostrar la aplicación práctica del modelo de datos relacional, en que en ese entonces se acababa de proponer. El nombre de SQL esta formado por las iniciales en inglés de "Structured Query Lenguaje" Lenguje Estructurado de Consulta. Todavía se le conoce con su antiguo nombre "siquel".

El lanzamiento de SQL tuvo un gran impacto en el ambiente. En Mayo de 1986, el Instituto Nacional Americano de Estandares (ANSI) declaró a SQL como el lenguaje estándar para bases de datos relacionales.

SQL por ser un lenguaje no procedural, permite que el usuario solicite a la computadora la información que desea ver y no el cómo la computadora obtendrá la información. La forma de construir una consulta a la base de datos se realiza con base en los comandos de SQL, los cuales son pocos y la facilidad que proporcionan al poder incrementar la complejidad de las consultas que se quieran realizar, hace que la tarea sea sencilla.

Beneficios de SQL

SQL permite que puedas trabajar con un nivel más alto de estructuras de datos. Más bien manipulando simples registros o conjuntos de registros. La forma más común de conjuntos de registros es una tabla. También todas las instrucciones de SQL aceptan conjuntos como entrada y todas las instrucciones regresan conjuntos como salida. SQL no requiere de un método específico de acceso a los datos. Esta característica hace de este más fácil para su concentración y obtención de resultados.

III. 2 ELEMENTOS DE SQL

Objetos de la base de datos

Los objetos de la base de datos son elementos que contiene ORACLE RDBMS. La mayoría de los objetos de la base de datos ocupan un espacio en la base de datos llamada tabla de espacios (tablespace) que sirve para guardar otros objetos en la base de datos. Las tablas son el único objeto de la base de datos que guarda los datos utilizados y estos se pueden accesar directamente.

Los objetos de la base de datos se definen con un nombre cuando se crean para identificarlos. Todos los objetos en la base de datos tienen un creador y un propietario; normalmente el creador y el propietario del objeto de la base de datos son el mismo.

La siguiente lista contiene nombres de objetos validos, que son:

- Column (columna)
- Databases (bases de datos)
- Indexes (indices)
- Rollback segments (segmentos de rollback)
- Synonyms (sinónimos)
- Tables (tablas)
- Tablespaces (tablas de espacios)
- Views (vistas)
- User (usuario)

Tipos de datos

Cada literal de la variable manipulada por la base de datos de ORACLE tiene un tipo de datos. Los tipos de datos son los siguientes:

CHAR

Este tipo incluye algunas letras del alfabeto, algunos numeros y algunos simbolos del teclado. Las literales CHAR deben ser encerradas entre apostrofes.

NUMBER

Este tipo incluye los digitos del 0 al 9, el punto decimal y el signo menos (-) si es necesario.Las literales NUMBER no son encerradas apostrofes.

DATE

Este tipo especial incluye información acerca de horas y dias. Este tiene el formato de default DD-MON-YY.

Valores Null (Nulos)

Si un renglón carece de un valor de un dato en una columna en particular, este valor puede ser null. Las columnas de algún tipo de dato puede ser que contengan valores nulos a menos que la columna este definida como NOT NULL cuando la tabla fue creada. Los valores nulos son apropiados cuando el valor actual es desconocido o cuando un valor debería no tener un significado. Además null no es equivalente al valor de cero.

Pseudo-columnas

Las pseudo-columnas se comportan como una columna de una tabla, pero no actuan al almacenarse en una tabla. Cuando se hacen consultas se puede hacer referencia a una pseudo-columna y estas no se pueden utilizar cuando se inserta, se borra o se actualiza.

Algunas pseudo-columnas son las siguientes:

- ROWID:
  Es un número que identifica a cada renglón de la tabla.

- ROWNUM:
  Este regresa un número indicando la secuencia en que el renglón ha sido seleccionado de la tabla o un conjunto de renglones.

- SYSDATE:
  Este regresa la fecha y hora actual.

Comandos de SQL

Los comandos de SQL estan diseñados para trabajar con la base de datos relacional y además pueden ser utilizados en otros productos de la compañía ORACLE (como SQL*Forms) , en utilerías (como SQL*DBA) y en programas (como PRO*C y PRO*COBOL).

Los comandos de SQL son divididos en 4 categorías basicamente, que son los siguientes:

1) Consultas (Querys):
   Las consultas son comandos que traen datos en alguna combinación, orden o expresión. Las consultas usualmente comienzan con una palabra SELECT de SQL, seguida por el dato deseado, y las tablas o vistas contenidas en la fuente de datos.

2) Manipulación de datos (DML):

Los comandos de DML son usados para combinar los datos por medio de los siguientes comandos:

- INSERT: insertar nuevos renglones de datos dentro de la tabla.
- DELETE: borrar renglones de la tabla.
- UPDATE: actualizar valores de las columnas en los renglones existentes.

3) Definición de datos (DDL):

Los comandos de DDL son usados para definir y mantener los objetos en la base de datos(incluyendo tablas y vistas) y para borrar estos cuando los objetos no son requeridos.

- CREATE: creación de objetos.
- ALTER: modificación de objetos.
- DROP: borrar objetos.

4) Control de datos(DCL):

Los comandos de DCL son usados para controlar los objetos.

- GRANT, CONNET: privilegios para accesar a la base de datos.
- REVOKE: eliminación de privilegios.

III.3 CONSULTAS Y MANIPULACION DE DATOS

Para entrar a SQL desde el prompt del sistema operativo tecleamos:

SQLPLUS <return>

Después tecleamos:

Enter username: nombre del usuario <return>
Enter password: clave de acceso <return>

Nota: El password no se ve.

Después para pedir ayuda sobre los comandos de SQL y SQL*PLUS tecleamos:

SQL> HELP COMMANDS

Y si queremos información de un solo comando tecleamos:

SQL> HELP SELECT
SQL> HELP CREATE

Si deseamos obtener la descripción de una tabla específica, utilizamos el comando DESCRIBE, la cual va a contener:

NAME nombre de la columna.
NULL? si se permiten valores nulos en ésta columna.
TYPE el tipo de dato (CHAR, NUMERIC, DATE)
Ejemplo:
SQL> DESCRIBE DEPTO

NAME	NULL?	TYPE
NUM_DPTO		NUMBER(2)
NO_DPTO		CHAR(14)
LOC		CHAR(13)

Sintáxis general de la consulta (Query)

Utilizando SQL, se puede consultar la base de datos de un número interminable de formas y su sintaxis general es:

SELECT	SELECT COL1, COL2,...,COLN
FROM	FROM NOMBRE DE LA TABLA
WHERE	WHERE CONDICION(ES)
GROUP BY .	ORDER BY COL1, COL2,...,COLN
HAVING...	
ORDER BY...	

Selección de todas las columnas

Si deseamos saber que datos hay en la tabla EMP, tecleamos lo siguiente:

```
SQL> SELECT * FROM EMP;
```

Nota: el asterisco significa todas las columnas.

NUM_EMP	NO_EMP	PUESTO	FEC_ING	SAL	COM	NUM_DPTO
7329	PEDRO	ANALISTA	17-JUN-92	1600	200	20
7449	ALEX	OPERADOR	20-FEB-92	800		20

Selección de una columna específica

Para seleccionar una columna, tecleamos lo siguiente:

```
SQL> SELECT NO_EMP
2 FROM EMP;

NOM_EMP
PEDRO
ALEX
SAUL
JORGE
```

Selección de múltiples columnas

Para seleccionar múltiples columnas, se ponen las columnas separadas por una coma cada una y aparecerán todos los renglones para cada columna seleccionada.

Ejemplo:

```
SQL> SELECT NUM_EMP, NO_EMP, PUESTO
2   FROM EMP;
```

NUM_EMP	NO_EMP	PUESTO
7329	PEDRO	ANALISTA
7449	ALEX	OPERADOR
7551	SAUL	CAPTURISTA
7560	JORGE	JEFE

Reordenación de columnas en la selección

El orden de selección determina el orden en que las columnas se despliegan.

Ejemplo:

```
SQL> SELECT PUESTO, NO_EMP, NUM_EMP
2   FROM EMP;
```

PUESTO	NO_EMP	NUM_EMP
ANALISTA	PEDRO	7329
OPERADOR	ALEX	7449
CAPTURISTA	SAUL	7551
JEFE	JORGE	7566

Selección de renglones

Si no se especifica una cláusula WHERE, se selecciona todos los regiones. Especificando una cláusula WHERE, se puede seleccionar ciertos renglones que cumplan con alguna o algunas condiciones.

Ejemplo:

```
SQL> SELECT NO_EMP
2   FROM EMP
3   WHERE NUM_DPTO = 20;

    NO_EMP
    PEDRO
    ALEX
    SAUL
    JORGE
```

Ordenando renglones

En el modelo relacional, los renglones no tienen un orden particular. La cláusula ORDER BY es la única forma como se puede asegurar que los renglones van a ser desplegados de acuerdo a cierto criterio .

Ejemplo:

```
SQL>  SELECT NO_EMP
2     FROM EMP
3     ORDER BY NO_EMP;

        NO_EMP
        ALEX
        JORGE
        PEDRO
        SAUL
```

Ordenando renglones con criterio múltiple

Cuando se ordena una consulta con criterio múltiple, se tiene lo siguiente:

- Orden primario es la primer columna listada.
- Orden secundario es la segunda columna listada.
- El default es el órden *ascendente* (ASC).
- Para ordenar en forma descendente, se agrega la palabra DESC después del nombre de la columna por la que se va a ordenar.

Ejemplo:

```
SQL> SELECT *
2   FROM DEPTO
3   ORDER BY NUM_DPTO DESC;

    NUM_DPTO    NO_DPTO      LOC
    40          FINANZAS     D.F
    20          INFORMATICA EDO. DE MEX
```

Seleccionando renglones especificos

Con la cláusula WHERE, se puede comparar el valor de una columna con:

- Una constante de carácteres usando apóstrofes.

Ejemplo:

WHERE NO_EMP = 'PEDRO'

Nota: se debe tomar en cuenta la forma como el dato está guardado en la base de datos respetando mayúsculas y minúsculas.

- Una expresión aritmética sin apóstrofes.

Ejemplo:

WHERE NUM_DPTO = 20

- El valor de una columna.

Ejemplo:

WHERE EMP.NUM_DPTO = DEPTO.NUM_DPTO

Operadores lógicos para la selección de renglones.

- Operadores de igualdad y desigualdad

Igual a	=
Desigual	! = ó <>
Mayor que	>
Mayor o igual	> =
Menor que	<
Menor o igual	< =

Otros operadores

Igual a cualquier elemento de la siguiente lista	IN(Lista)
Mayor o igual a un valor menor o igual que otro	BETWEEN menor AND mayor
Coincide con un patrón	LIKE
Una cadena de cero o más caracteres	%
Una cadena de un carácter	_
No aplicable o	

valor inexistente	IS NULL
Negador de algunos operadores (Ej. NOT IN, IS NOT NULL, etc.)	NOT

Lista de valores

El operador IN permite seleccionar valores que coincidan con uno de los valores en la lista.
Ejemplo:
Qué empleados son, ya sea analista o capturista?

```
SQL> SELECT NO_EMP, PUESTO
2   FROM EMP
3   WHERE PUESTO IN ('ANALISTA', 'CAPTURISTA');
```

NO_EMP	PUESTO
PEDRO	ANALISTA
SAUL	CAPTURISTA

El operador NOT IN permite seleccionar los renglones que no caen en la lista.
Ejemplo:

```
SQL> SELECT NO_EMP, PUESTO
2   FROM EMP
3   WHERE  TRABAJO NOT IN ('ANALISTA', 'CAPTURISTA');
```

NOM_EMP	PUESTO
ALEX	OPERADOR
JORGE	JEFE

Rango de valores

El operador BETWEEN permite seleccionar renglones que contengan valores dentro de un rango.

Ejemplo:
Qué empleados ganan entre 500 y 1500?

```
SQL> SELECT NO_EMP, PUESTO, SAL
2  FROM  EMP
3  WHERE SAL BETWEEN 500 AND 1500;
```

NO_EMP	PUESTO	SAL
ALEX	OPERADOR	800
SAUL	CAPTURISTA	1250

El operador NOT BETWEEN selecciona renglones fuera de un rango.
Ejemplo:

```
SQL> SELECT NO_EMP, PUESTO, SAL
2    FROM EMP
3    WHERE SAL NOT BETWEEN 500 AND 1500;
```

NO_EMP	PUESTO	SAL
PEDRO	ANALISTA	1600
JORGE	JEFE	2975

Busqueda de patrones

Para buscar una cadena de caracteres, se utiliza el operador LIKE en la cláusula WHERE.

Ejemplos:
Listar a todos los empleados cuyos nombres empiecen con P?

```
SQL> SELECT NO_EMP, NUM_DPTO
2  FROM EMP
3  WHERE NOMBRE LIKE 'P%';
```

NO_EMP	NUM_EMP
PEDRO	20

Lista a todos los empleados cuyos nombre terminen con E?

```
SQL> SELECT NO_EMP, NUM_DPTO
2   FROM EMP
3   WHERE NO_EMP LIKE '%E';

   NO_EMP      NUM_DPTO
   JORGE       20
```

Otro ejemplo sería utilizando NOT LIKE para seleccionar renglones que no coincidan con un patrón.

Listar a todos los empleados cuyos nombre no empiecen con la cadena AL?

```
SQL> SELECT NO_EMP, NUM_DPTO
2   FROM EMP
3   WHERE NO_EMP NOT LIKE 'AL%';

   NO_EMP      NUM_DPTO
   PEDRO       20
   JORGE       20
```

Valores nulos (NULL)

Un valor nulo en una columna no es lo mismo que un cero, porque cero es un número y NULL no es un número. Por lo tanto, NULL significa que el valor es desconocido, faltante o no aplicable.

Ejemplo:

Listar a todos los empleados que no son candidatos para recibir comisión (cuyo valor en la columna COM sea nulo).

```
SQL> SELECT NO_EMP, PUESTO
2    FROM EMP
3    WHERE COM IS NULL;

   NO_EMP      PUESTO
   ALEX        OPERADOR
```

SAUL CAPTURISTA

Para buscar valores no nulos.

Ejemplo:

```
SQL> SELECT NO_EMP, PUESTO
2  FROM  EMP
3  WHERE COM IS NOT NULL;

   NO_EMP      PUESTO
   ALEX        ANALISTA
   SAUL        JEFE
```

Condiciones múltiples

La cláusula WHERE puede calificar y seleccionar renglones de las tablas especificando más de una condición de busqueda.

Ejemplo:

```
SQL> SELECT NO_EMP, PUESTO
2  FROM  EMP
3  WHERE NUM_DPTO = 20    AND PUESTO != 'ANALISTA';

   NO_EMP      PUESTO
   ALEX        OPERADOR
   SAUL        CAPTURISTA
   JORGE       JEFE
```

El operador AND agrega el criterio adicional que las columnas seleccionadas deben cumplir.

El operador OR especifica selección de renglones que cumplan cualquiera de las condiciones (una o la otra).

Ejemplo:

```
SQL> SELECT NO_EMP, PUESTO
```

```
2  FROM EMP
3  WHERE NUM_DPTO = 20
4  OR PUESTO != 'OPERADOR';

   NO_EMP      PUESTO
   PEDRO       ANALISTA
   ALEX        OPERADOR
   SAUL        CAPTURISTA
   JORGE       JEFE
```

Inserción de datos

Para insertar datos en una tabla , es necesario que la tabla este creada, los valores se separan por comas y si es necesario utilizar el comando DESCRIBE para ver el orden y tipo de dato de las columnas de la tabla. Los valores deben coincidir con el tipo de dato de la columna a la que se insertan. Los datos tipo CHAR y DATE deben ponerse entre apóstrofes.

Se pueden listar los nombres de las columnas al insertar para:

- Insertar datos solamente en algunas columnas de la tabla.
- Introducir los datos en alguna secuencia especifica.

Los valores se deben dar en el orden especificado en la cláusula INSERT.
Ejemplo:

```
SQL> INSERT INTO DEPTO
2  VALUES(10,'CONTADURIA','GUADALAJARA');

      NUM_DPTO   NO_DPTO      LOC
      10         CONTADURIA   GUADALAJARA
```

Insertando renglones de otra tabla

Se puede utilizar el comando INSERT con una consulta (query) para seleccionar renglones de una tabla e insertarlos en otra. La consulta sustituye la cláusula VALUES.
Ejemplo:

```
SQL> INSERT INTO EMP (NUM_EMP, NO_EMP, NUM_DPTO)
2     SELECT ID, NOMBRE, DEPARTAMENTO
3     FROM EMP1
4     WHERE DEPARTAMENTO IN (10,20,30);
```

Utilizando parámetros en el comando INSERT

El comando INSERT puede contener parámetros representando valores que van a ser provistos por el usuario al momento de correr el comando. Cada parámetro consiste de un & seguido por el nombre de la columna.

No es necesario poner apóstrofes en datos tipo CHAR y DATE si se ponen en el parámetro.

Ejemplo:

```
SQL> INSERT INTO DEPTO
2     VALUES(&NUM_DPTO, &NO_DPTO, &LOC);
```

Insertando valores nulos

Si no se incluye una columna de la tabla en la cláusula INSERT, el valor de esa columna queda como nulo por default. Se puede especificar NULL en la cláusula VALUES (a menos que NOT NULL esté especificado para esa columna).

Ejemplo:

```
SQL> INSERT INTO DEPTO
2     VALUES (50, 'TELECOM', NULL);
```

Insertando valores tipo DATE.

El formato default para introducir fechas es:

'DD-MON-YY'

Ejemplo:

```
SQL> INSERT INTO EMP(NUM_EMP,NO_EMP, FEC_ING)
2  VALUES(7964, 'JOEL', '07-APR-92');
```

Para introducir automáticamente la fecha y hora actual, se utiliza :

SYSDATE.

Ejemplo:

```
SQL> INSERT INTO EMP(NUM_EMP, NO_EMP, FEC_ING)
2    VALUES(7600, 'ROBERTO',SYSDATE);
```

Actualización de datos

Para actualizar datos se utiliza el comando UPDATE.

Ejemplo:

Se requiere promover a el Sr. Jorge como Gerente.

```
SQL> UPDATE EMP
2    SET PUESTO = 'GERENTE'
3    WHERE NOMBRE = 'JORGE';
```

TABLA ORIGINAL

NUM_EMP	NO_EMP	PUESTO	FEC_ING	SAL	COM	NUM_DPTO
7329	PEDRO	ANALISTA	17-JUN-92	1600	200	20
7449	ALEX	OPERADOR	20-FEB-92	800	20	
7521	SAUL	CAPTURISTA	24-OCT-91	1250	20	
7566	JORGE	JEFE	02-APR-90	2975	300	20

TABLA ACTUALIZADA

NUM_EMP	NO_EMP	PUESTO	FEC_ING	SAL	COM	NUM_DPTO
7329	PEDRO	ANALISTA	17-JUN-92	1600	200	20
7449	ALEX	OPERADOR	20-FEB-92	1200	20	
7521	SAUL	CAPTURISTA	24-OCT-91	1200	20	
7566	JORGE	GERENTE	02-APR-90	2975	300	20

Nota: si se omite la cláusula WHERE, todos los valores en la columna cambiarán al valor en la cláusula SET.

Actualizando múltiples columnas

Se pueden actualizar múltiples columnas especificando los cambios en la cláusula SET.
Ejemplo:

```
SQL> UPDATE DEPTO
2   SET NUM-DPTO = 30,  NO_DPTO = 'PERSONAL';
3   WHERE NUM_DPTO = 40;
```

Borrado de renglones

No se puede borrar renglones parciales, por lo que si se desea hacer esto, actualice la columna por NULL.

La cláusula WHERE determina los renglones a borrar de la tabla y si se omite esta cláusula se borran todos los renglones de la tabla.

Ejemplo:

```
SQL> DELETE FROM EMP
2       WHERE NUM_EMP = 7449;
```

Para hacer cambios permanentes

Cuando se inserta, se borra o se actualiza una tabla no se hacen permanentes los cambios hasta que se hace un COMMIT a la base de datos.

Existen varias formas de hacer un COMMIT, que son las siguientes:

a) COMMIT explícito: se debe dar el comando COMMIT de SQL para que los cambios se hagan permanentes.

```
SQL> COMMIT;
```

b) COMMIT implícito: los siguientes comandos de SQL causan un COMMIT implícito:

ALTER, CONNECT, CREATE, DISCONNECT, DROP, GRANT, REVOKE, RENAME

c) COMMIT automático: los cambios tienen efecto inmediatamente después de un INSERT, UPDATE o DELETE si el AUTOCOMMIT se encuentra habilitado.

Para esto se utiliza el commando SET de SQL*PLUS:

SQL> SET AUTOCOMMIT ON

Recuperando el último cambio

El comando ROLLBACK cancela todos cambios pendientes regresando al estado en que estaba la información al momento del último COMMIT.

SQL> ROLLBACK;

Transacciones lógicas

Todos los cambios a la base de datos entre un COMMIT y otro, se conocen commo una transacción. Cuando una transacción es interrumpida por un error serio, como una falla del sistema, se hace un ROLLBACK de toda la transacción. Esto previene el error de que sólo una parte de la transacción lógica quede salvada en la base base de datos.

III.4 EXPRESIONES Y FUNCIONES

Se puede utilizar SQL como una calculadora para expresar valores con operadores aritméticos, tales como:

- \+ SUMA
- \- RESTA
- * MULTIPLICACION
- / DIVISION

Estos operadores pueden ser utilizados en cualquiera de las siguientes cláusulas:

SELECT
WHERE
ORDER BY

HAVING

Expresiones numéricas

Se puede utilizar más de una expresión aritmética en una consulta (query).

Ejemplo:

Hacer una consulta para saber quién gana comisiones que representen más del 5% de su salario.

```
SQL> SELECT NO_EMP, SAL, COM, COM/SAL
2    FROM EMP
3    WHERE COM> 0.5*SAL
4    ORDER BY COM/SAL DESC;

NO_EMP   SAL   COM   COM/SAL
PEDRO    1600  200   8
JORGE    2975  300   9.9
```

Expresiones numericas con operadores múltiples

Es posible anidar varias expresiones, a continuación se presenta el orden de evaluación:

- Se evalua primero:
  - multiplicación *
  - división /
- Después se evalua:
  - suma +
  - resta -

La evaluación de expresiones se efectúa de izquierda a derecha en donde el orden de evaluación anterior no se aplica y se pueda controlar el orden de evaluación utilizando (), por ejemplo:

```
12*(SAL +COM) != 12*SAL + COM
```

Uso de expresiones aritméticas en FECHAS

Se puede utilizar expresiones para especificar valores que involucren datos tipo DATE. Esta expresión aritmética utilizada a los días como unidad básica. Y es posible sumar y restar fechas.

Algunos ejemplos utilizando suma de fechas son:

- Sumar 2 días al 6-MAR-87
  6-MAR-87 + 2 = 8-MAR-87

- Sumar 2 horas al 6-MAR-87
  6-MAR-87 +2/24 = 6-MAR-87 y 2 horas

- Sumar 15 segundos al 6-MAR-87
  6-MAR-87 + 15/(24*60*60) = 6-MAR-87 y 15 segs.

Utilizando alias de columnas

El encabezado de las columnas que se despliega normalmente refleja el nombre de la columna que usted especifica al crear la tabla.

Se puede cambiar el encabezado de la columna que se despliega especificando un alias de columna en la cláusula SELECT.

para especificar un alias, se debe dejar un espacio en blanco después del nombre de la columna y después el alias que usted quiera colocar.

Ejemplo:

```
SQL> SELECT NO_EMP EMPLEADO
2     FROM EMP
3     WHERE NUM_DPTO = 10;

   EMPLEADO
   PEDRO
   ALEX
   SAUL
   JORGE
```

Alias de columnas y expresiones

Si una cláusula SELECT contiene una expresión aritmética, esta expresión es utilizada como el encabezado de la columna. Un alias se puede utilizar para renombrar temporalmente una columna calculada para hacer que los resultados de una consulta sean mas legibles.

La expresión, no el alias, es la que se refiere en la cláusula ORDER BY. Esta regla se aplica también para todas las otras cláusulas en el SELECT.

También un alias que contenga caracteres especiales como espacio o /, debe ponerse entre comillas.

Ejemplo:

```
SQL> SELECT NO_EMP, SAL, COM, COM/SAL "C/S RANGO"
2  FROM EMP
3  WHERE COM> 0.05 *SAL
4  ORDER BY COM/SAL DESC;

NO_EMP   SAL   COM   C/S RANGO
PEDRO    1600  200   8
JORGE    2975  300   9.9
```

Propósito de la funciones

Los propósitos de las funciones son:

- Modificar valores.
- Modificar valores para crear nuevos valores.
- Cambiar formatos de valores.

Como una expresión, una función puede ser utilizada en las cláusulas siguientes:

SELECT, WHERE, ORDER BY, HAVING

Si no se utiliza como expresión, una función tiene el siguiente formato:

FUNCION(argumento)

también puede tener múltiples argumentos

FUNCION(arg1, arg2,...,argn)

El primer argumento de una función en SQL es siempre el valor al cual se le aplica la función.

Existen varios tipos de funciones:

1) Clasificadas según su tipo de valor

- Funciones aritméticas.
- Funciones CHAR.
- Funciones DATE

2) Clasificadas según los renglones que afectan.

- Funciones individuales:
  Cada renglón es evaluado por separado, la función se aplica a cada renglón de la tabla.
- Funciones de grupo:
  Se evalúan colectivamente conjuntos de renglones.

La ventaja principal de una función es la de poder regresar un valor basado en múltiples argumentos.

Funciones tipo CHAR más comunes

En total existen 15 funciones de tipo CHAR, pero las más comunes son las siguientes:

INITCAP(NO_EMP) pone en mayúsculas la primera letra de cada palabra.

pedro se vuelve Pedro

LENGTH(NO_EMP) calcula el número de caracteres en la cadena.

length de Alberto es 7

SUBSTR(PUESTO,1,4) lista 4 caracteres comenzando con el primer carácter de la cadena.

Jorge se vuelve Jorg

LOWER convierte todos los caracteres de la cadena a minúsculas.

UPPER convierte todos los caracteres de la cadena en mayúsculas.

LEAST regresa el valor, de una serie de argumentos que esté primero en orden alfabético.

GREATEST regresa el valor, de una serie de argumentos que este al final en orden alfabético.

Funciones tipo DATE

Las funciones tipo DATE se especifican en la misma forma que las funciones tipo CHAR, con el nombre de la función seguido por sus argumentos.

A continuación se presentan 3 funciones simples de tipo DATE:

1)ADD_MONTHS(FEC_ING,5):
Suma 5 meses a FEC_ING.

2)MONTHS_BETWEEN(SYSDATE, FEC_ING):
Calcula el número de meses entre FEC_ING y SYSDATE.

3)NEXT_DAY(FEC_ING,'FRIDAY'):
Encuentra la fecha del siguiente viernes a partir de la fecha de ingreso (FEC_ING).

Ejemplo:
Mostrar que salarios serán pagados en el departamento 20 el próximo viernes.

```
SQL> SELECT NO_EMP, SAL, NEXT_DAY(SYSDATE, 'FRIDAY') PROX_V
2    FROM EMP
3    WHERE NUM_DPTO = 20;

NO_EMP   SAL    PROX_V
```

```
PEDRO    1600   30-NOV-92
ALEX     800    30-NOV-92
SAUL     1250   30-NOV-92
JORGE    2975   30-NOV-92
```

La función TO_CHAR

Las fechas son despliegas en el formato default de ORACLE, a menos que se utilice la función TO_CHAR.

La forma es:

TO_CHAR(DATE, date picture)

La fecha va a ser representada como una cadena de caracteres de acuerdo al formato dado en el date picture.

Ejemplo:

```
SQL> SELECT NO_EMP, TO_CHAR(FEC_ING,'DyMondd,yyyy') INGRESO
2   FROM EMP
3   WHERE NUM_DPTO = 20;

NO_EMP   INGRESO
PEDRO    Tue Jun 17, 1992
ALEX     Sat Feb 20, 1992
SAUL     Fri Oct 24, 1991
JORGE    Wes Apr 02, 1990
```

La función TO_DATE

La función TO_DATE convierte una cadena de caracteres de una gran variedad de formatos a un dato tipo DATE de ORACLE.

La forma es:

TO_DATE(cadena de caracteres,date picture)

La cadena de caracteres va a ser convertida a una fecha ORACLE de acuerdo al formato especificado en el date picture.

Date pictures

Cuando se utiliza un "date picture" debe ponerlo entre apóstrofes.
Algunos date pictures son:

Dia		
dd	número	12
dy	abreviado	fri
day	deletreado	friday

Mes		
mm	número	03
mon	abreviado	mar
month	deletreado	march

Año		
yy	año	87
yyyy	año y siglo	1990

Nota: si pone en mayúsculas o abrevia el date picture, la salida lo refleja.

Ejemplos:

'Mar 12, 1988'	'Mon dd, yyyy'
'MAR 12, 1988'	'MON dd, yyyy'
'Thurday MARCH 12'	'DAY MONTH dd'
'Mar 12 11:00 am'	'Mon dd hh:mi am'

Funciones numéricas

Las funciones numéricas se especifican de la misma forma que las funciones CHAR y DATE, con el nombre de la función seguido por sus argumentos entre paréntesis.

Utilizando funciones numéricas, se puede efectuar operaciones sobre un conjunto de valores de una tabla o vista.

ABS(COM-SAL)
Retorna el valor absoluto de la diferencia entre COM y SAL.

GREATEST(SAL,COM)
Retorna el valor mayor (SAL o COM)

ROUND(SAL,0)
Redondea SAL al dólar más cercano (sin decimales).

SIGN(COM-SAL)
Puede regresar los valores siguientes:

sign = -1 si COM-SAL<0
sign = 0 si COM-SAL=0
sign = 1 si COM-SAL>0

La función NVL

Permite un manejo apropiado de valores nulos (sin NVL, cualquier número más un dato tipo NULL me da un resultado NULL).
La forma es:

NVL(arg1,arg2)

donde:
arg1 es el nombre de la columna. Si arg1 es no NULL, NVL trae su valor. Y si el arg1 es NULL, NVL regresa el valor de arg2.

Ejemplo:
Cuánto recibe en total (salario más comisión) cada empleado del departamento 20.

```
SQL> SELECT NO_EMP, COM+,  NVL(COM,0) + SAL
2       FROM EMP;
```

NO_EMP	COM+SAL	NVL(COM,0) + SAL
PEDRO	1800	1800
ALEJANDRO	800	
ALBERTO	1200	
JORGE	3275	3275

Funciones de grupo

En general, las funciones de grupo regresan un valor único para un conjunto de renglones y se pueden aplicar cualquier valor numérico y algunos valores tipo CHAR y DATE.

Ejemplo:
Encontrar el total de comisiones pagadas a los empleados

```
SQL> SELECT SUM(COM)
2     FROM EMP;

    SUM(COM)
       500
```

Por otro lado, no se puede seleccionar resultados individuales y de grupo en el mismo query a menos que contruya una cláusula GROUP BY.

Ejemplo:

```
SQL> SELECT NO_EMP, SUM(COM)
2     FROM EMP;
```

Error . . . not a single group set function

También existen funciones de grupo para cualquier tipo de dato(NUMBER, CHAR o DATE), estas funciones son MIN, MAX y COUNT.

Ejemplos:

```
SQL> SELECT MIN(NO_EMP)
2     FROM EMP;
```

```
COUNT(DISTINCT PUESTO)
4
```

Más funciones de grupo(para valores numéricos)

La función SUM obtiene la suma.

Ejemplo:
Encontar el total de los salarios pagados.

```
SQL> SELECT SUM(SAL)
2  FROM EMP;

   SUM(SAL)
   6625
```

La cláusula GROUP BY

Se utiliza la cláusula GROUP BY para definir múltiples grupos de renglones. Cada miembro del grupo tiene por lo menos un valor en común.

Se deben especificar las columnas que contengan estos valores comunes en la cláusula GROUP BY.

Ejemplo:

```
SQL> SELECT NO_DPTO; SUM(SAL)
2      FROM EMP;
3      GROUP BY NO_DPTO;
```

Nota: una consulta con la cláusula GROUP BY regresa un renglón por grupo.

Criterios multiples para agrupar renglones

se puede agrupar por más de una columna.

Ejemplo:

```
   MIN(NO_EMP)
   ALBERTO

SQL> SELECT MIN(FEC_ING)
2     FROM EMP;

   MIN(FEC_ING)
   02-APR-90

SQL> SELECT MIN(SAL)
2     FROM EMP;

   MIN(SAL)
   800
```

Las funciones MAX y COUNT funcionan de la misma manera.

La función COUNT DISTINCT

La función COUNT DISTINCT obtiene el número de valores únicos en la columna.

Ejemplo:
Cuantos empleados tienen puestos?

```
SQL> SELECT COUNT(PUESTO)
2     FROM EMP;

   COUNT(PUESTO)
   4
```

Ejemplo:
Cuantos puestos diferentes existen?

```
SQL> SELECT COUNT(DISTINCT PUESTO)
2     FROM EMP;
```

```
SQL> SELECT NO_DPTO, PUESTO, COUNT(*)
2    FROM EMP
3    GROUP BY NO_DPTO, PUESTO
```

Seleccionando grupos específicos

La cláusula HAVING permite seleccionar grupos que cumplan con cierta(s) condición(es) y es análoga a la cláusula WHERE, pero que sirve para un propósito diferente.

- La cláusula WHERE pone condiciones sobre la cláusula SELECT.

- La cláusula HAVING pone condiciones sobre la cláusula GROUP BY.

Ejemplo:
Qué departamentos tienen una nómina de más de 9,000 (no incluir comisiones)?

```
SQL> SELECT NO_DPTO, SUM(SAL)
2  FROM EMP
3  GROUP BY NO_DPTO
4  HAVING SUM(SAL)>9000;
```

III.5 CREACION DE TABLAS Y VISTAS

Crear una tabla

Cuando se crea una tabla, se debe especificar:

- Nombre de la tabla.
- Nombres de las columnas.
- Tipos de valores de la columnas.
- Longitud máxima de las columnas.

Nota: una tabla puede contener hasta 254 columnas.

Ejemplo:

```
SQL> CREATE TABLE DEPTO
2   (NUM_DPTO   NUMBER(2),
3   NO_DPTO     CHAR(14),
4   LOC         CHAR(13));
```

Reglas para nombrar tablas y columnas

1) Restricciones en nombres

- El primer caracter debe ser alfabético (A-Z o a-z) pero todos se guardan con mayúsculas.
- Los caracteres subsiguientes pueden ser números, $, #, y _ (no se permiten comas).
- Los nombres pueden ser hasta de 30 caracteres.

2) Los nombres deben ser únicos

- El nombre de su tabla debe ser único dentro de su cuenta.
- No se pueden utilizar palabras reservadas de ORACLE.
- Los nombres de las columnas deben ser únicos dentro de la tablas.

3) Se pueden utilizar comillas

- Si ponemos el nombre de la tabla entre comillas, no se aplican las reglas anteriores en cuanto a caracteres.
- El utilizar mayúsculas o minúsculas solo hacen diferencia cuando usamos comillas.
- Todos los accesos subsecuentes a los objetos nombrados de esta forma, requieren también del uso de comillas.

Prohibiendo valores nulos

A veces se quiere asegurar que una columna no tenga valores nulos, por ejemplo, puede querer evitar que se inserte un renglón a la tabla de empleados (EMP) si no se conoce el número de empleado (NUM_EMP).

Para prohibir que se inserten valores nulos en una columna, se debe poner la cláusula NOT NULL después de la información de la columna. A partir de entonces, cada vez que se quiera

insertar un renglón sin valor para una columna que se definió como NOT NULL, va a mandar un mensaje de error y la operación va a fallar.

Ejemplo:

```
SQL> CREATE TABLE DEPTO
2  (NUM_DPTO     NUMBER(2) NOT NULL,
3   NO_DPTO      CHAR(14),
4   LOC          CHAR(13));
```

Agregando una nueva columna a una tabla

Cuando una columna es agregada a una tabla, todos sus renglones son NULL.

Ejemplo:

```
SQL> ALTER TABLE DEPTO
2    ADD (ZONA    NUMBER(3));
```

Tabla original

NUM_DPTO	NO_DPTO	LOC
40	FINANZAS	D.F
20	INFORMATICA	EDO. DE MEX

Tabla alterada

NUM_DPTO	NO_DPTO	LOC	ZONA
40	FINANZAS	D.F	
20	INFORMATICA	EDO. DE MEX	

Alargando una columna

- Se utiliza el commando ALTER para cambiar la longitud de la columna.
- No se puede reducir ni cambiar el tipo de la columna a menos de que la columna este vacía.
- No se puede modificar la columna y hacerla NOT NULL a menos que todos los renglones tengan valores para esa columna.

Ejemplo:

```
SQL> ALTER TABLE DEPTO
2   MODIFY (NO_DPTO  CHAR(20));
```

Vistas

Una vista es como una ventana a través de la cual se puede consultar o modificar información de una tabla.

Una vista es una tabla virtual, porque:

- Se ve como una tabla, pero no existe como tal.
- Sus datos se derivan de las tablas.
- A pesar de esto, no hay copia de los datos.

Una vista provee:

- Simplicidad para ver exactamente lo que se necesita.
- Seguridad para prevenir que los usuarios no autorizados vean ciertas columnas o renglones de las tablas.

Nota: se pueden tener varias vistas de la misma tabla.

Crear, nombrar y consultar vistas

Para crear una vista es diferente que crear una tabla y se puede utilizar cualquier SELECT que no contenga un ORDER BY cuando se crea una vista.

Ejemplo:

```
SQL> CREATE VIEW ANALISTAS AS
2   SELECT NO_EMP, PUESTO, SAL
3   FROM EMP
4   WHERE PUESTO = 'ANALISTA';
```

Para consultar una vista, se hace un SELECT como si la vista fuera una tabla.

Ejemplo:

```
SQL> SELECT *
2     FROM ANALISTAS;
```

Creando vistas con alias de nombres de columnas

A menos que se especifique lo contrario, la vista hereda los nombres de las columnas de la tabla de la que se deriva.

Para dar a la vista nombres de columnas diferentes a los de la tabla base, se utiliza la siguiente sintaxis:

```
CREATE VIEW nombre_vista (alias, alias, ...)
AS query;
```

Ejemplo:

```
SQL> CREATE VIEW MI_DEPT
2  (PERSONAS, TITULOS, SALARIO)
3   AS SELECT ENAME, JOB, SAL
4   WHERE NUM_DPTO = 20;
```

Más acerca de las vistas

La cláusula WITH CHECK OPTION especifica que las inserciones y actualizaciones hechas a la base de datos a partir de la vista, no pueden manipular datos que la vista no pueda seleccionar.

Ejemplo:

```
SQL> CREATE VIEW DEPT20 AS
2  SELECT NO_EMP, PUESTO , SAL, NUM_DPTO
3  FROM EMP
4  WHERE NUM_DPTO = 20
```

5 WITH CHECK OPTION;

Copiando tablas y vistas

Las razones para copiar tablas y vistas son:

- Respaldar tablas y vistas originales.
- Dar una copia a alguien mas.
- Hacer cambios tentativos.
- Archivarlas antes de borrarlas.

Para copiar una tabla se utiliza el comando CREATE TABLE con la cláusula AS seguida por un subquery.

Cuando la tabla es copiada, la copia incluye la definición de las columnas y los datos.

Borrando tablas y vistas

Para borrar tablas se utiliza el comando:

DROP TABLE nombre_tabla;

Si la tabla tiene datos, estos van a ser borrados permanentemente y una vez que se borra la tabla, no se pueden recuperar los datos.

Para borrar vistas se utiliza el comando:

DROP VIEW nombre_vista;

Nota: las tablas en las que se basa la vista no se alteran.

III.6 CONSULTAS AVANZADAS

Conceptos básicos de Joins

Los Joins se utilizan para combinar en una misma consulta (query), columnas de diferentes tablas.

A continuación se muestran algunos tipos de Joins.

Join simple (Equijoin)

En el WHERE se especifican las columnas sobre las que se lleva a cabo el Join, es decir, las columnas que tienen en común las tablas.

Ejemplo:

```
SQL> SELECT NUM_EMP, NO_EMP, PUESTO, DEPTO.NO_DPTO
2     FROM EMP, DEPTO
3     WHERE EMP.NUM_DPTO = DEPTO.NUM_DPTO;
```

Tablas EMP y DEPTO relacionadas

NUM_EMP	NO_EMP	PUESTO	NO_DPTO
7329	PEDRO	ANALISTA	INFORMATICA
7449	ALEX	OPERADOR	INFORMATICA
7521	SAUL	CAPTURISTA	INFORMATICA
7566	JORGE	JEFE	INFORMATICA

Nota: cuando se olvida la cláusula WHERE, cada renglón es relacionado con todos los renglones de la otra tabla y el resultado es un producto cartesiano.

Joins externos (Outer Joins)

Para desplegar renglones de una tabla que no tiene correspondencia en otra, es necesario el operador de outer join que es un signo "+" entre paréntesis.

Ejemplo:

```
SQL> SELECT NO_EMP, DEPTO.NUM_DPTO, LOC
2     FROM EMP, DEPTO
3     WHERE EMP.NUM_DPTO(+) = DEPTO.NUM_DPTO;
```

NO_EMP	NUM_DPTO	LOC
PEDRO	20	D.F.
ALEX	20	EDO. DE MEX
SAUL	20	EDO. DE MEX
JORGE	20	EDO. DE MEX
	40	EDO. DE MEX

Si existen valores en DEPTO.NUM_DPTO que no tengan correspondientes en EMP.NUM_DPTO, el simbolo de Join externo (+) causa que aparezca un valor nulo en la tabla relacionada.

Join de tablas consigo mismas (Self Join)

Cuando se requiere relacionar un renglón de la tabla con otro renglón de la misma tabla, se lleva a cabo un Self Join.

Ejemplo:

```
SQL > SELECT WORKER.NO_EMP, JOB.PUESTO
2     FROM EMP WORKER, EMP JOB
3     WHERE WORKER.NUM_DPTO = JOB. PUESTO
```

NO_EMP	PUESTO
PEDRO	ANALISTA
ALEX	OPERADOR
SAUL	CAPTURISTA
JORGE	JEFE

Se relaciona una tabla consigo mismo en el WHERE, como si fueran 2 tablas separadas.
Es necesario usar un alias por lo menos para una ocurrencia de la tabla en la cláusula FROM para poder diferenciar los nombres de las columnas según la tabla a la que pertenecen.

Join de No-Equivalencia (Non-Equijoin)

Si la condición de join en la cláusula WHERE especifica una equvalencia (=), es un Equijoin.
Una condición de Join con cualquier otro operador es un Non-Equijoin.Ejemplo:

Utilizando una nueva tabla

Tabla: GRADOSAL

GRADO	SAL_BAJ	SAL_ALT
1	500	1000
2	1001	1500
3	1501	2000
4	2001	2500
5	2501	3000

Qué salarios caen en el grado 3?

```
SQL> SELECT NO_EMP, SAL
2  FROM EMP, GRADOSAL
3  WHERE GRADO = 3
4  AND SAL BETWEEN SAL_BAJ AND SAL_ALT;
```

NOM_EMP	SAL
SAUL	1250

Operadores de conjuntos

Los operadores de conjuntos combinan 2 o más consultas (querys) en un resultado.

Operador UNION

La UNION regresa todos los distintos renglones seleccionados por cualquier consulta a los que se aplica y combina renglones de múltiples tablas y vistas.

Tabla 1

Tabla 1

UNION

Tabla 2 Tabla 2

Teniendo una vista EMP1

NO_EMP	SAL	PUESTO
RAUL	2100	ANALISTA
CARLOS	500	AYUDANTE GRAL

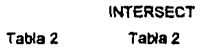
Ejemplo:

```
SQL> SELECT NO_EMP, SAL
2     FROM EMP
3     WHERE SAL>2000
4     UNION
5     SELECT NOM_EMP
6     FROM EMP1
7     WHERE SAL>2000;
```

NO_EMP	SAL	PUESTO
JORGE	2975	JEFE
RAUL	2100	ANALISTA

Operador INTERSECT

El operador INTERSECT encuentar renglones comunes en múltiples tablas y las columnas deben ser del mismo tipo.



Ejemplo:

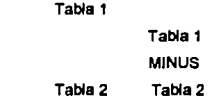
Qué puestos tienen en común todos los departamentos?

```
SQL> SELECT PUESTO
```

```
2    FROM EMP
3    INTERSECT
4    SELECT PUESTO
5    FROM EMP1;
```

Operador MINUS

El operador MINUS regresa los renglones que sean únicos para la primer consulta.



Ejemplo:

Existe algún puesto en el que el departamento de la tabla EMP1 no se encuentre en el departamento de la tabla EMP?

```
SQL> SELECT PUESTO
2    FROM EMP
3    MINUS
4    SELECT PUESTO
5    FROM EMP1;

PUESTO
AYUDANTE GRAL
```

Subconsultas (Subquerys)

Un subquery (query anidado) es un query que se encuentra dentro de la cláusula WHERE. Los resultados del subquery se utilizan para resolver el query principal.

```
          SELECT . . .
QUERY     FROM  . . .
PRINCIPAL WHERE . . .
```

```
SELECT . . .
        FROM . . .
        WHERE . . .

        SUBQUERY
```

Ejemplo:

Qué empleados trabajan en el mismo departamento que SAUL?

Solución:

Paso 1: hacer un subquery para encontrar el departamento en el que trabaja SAUL.

Paso 2: hacer un query principal para seleccionar a todos los empleados que trabajan en ese departamento.

```
SQL> SELECT NO_EMP, NUM_DPTO
2       FROM EMP
3       WHERE NUM_DPTO =
4       (SELECT NUM_DPTO FROM EMP
5       WHERE NO_EMP = 'SAUL');
```

Los subquerys pueden tener varios niveles y el anidamiento de los subquerys pueden continuar indefinidamente.

El subquery puede accesar tablas que no son utilizadas por el query principal.

Múltiples subquerys

Un subquery puede construir una de las partes de cualquier operador relacional.

Ejemplo:

Qué empleados tienen el mismo puesto que ALEX o un salario mayor al de él?

```
SQL> SELECT NO_EMP, PUESTO, SAL
2       FROM EMP
3       WHERE PUESTO =
```

```
4     (SELECT PUESTO FROM EMP
5     WHERE NO_EMP = 'ALEX');
6     OR SAL>
7      (SELECT SAL FROM EMP
8      WHERE NO_EMP = 'ALEX');
```

NO_EMP	PUESTO	SAL
SAUL	CAPTURISTA	250
JORGE	JEFE	2975

Múltiples tablas y subquerys

Se puede extraer información de más de una tabla en un query que contenga subquerys.

Ejemplo:
Qué empleados en el EDO DE MEX tienen un salario mayor al de SAUL?

```
SQL> SELECT NOM_EMP, PUESTO, SAL
2     FROM EMP, DEPTO
3     WHERE LOC = 'EDO DE MEX'
4     AND SAL >
5     (SELECT SAL FROM EMP
6     WHERE NOM_EMP = 'SAUL');
```

NOM_EMP	PUESTO	SAL
JORGE	JEFE	2975

Funciones de grupo en subquerys

Si selecciona una columna regular y una función de grupo en el mismo SELECT, recibe un mensaje de error.

Ejemplo:
Cuál fue el primer empleado contratado?

```
SQL> SELECT NOM_EMP, MIN(FEC_ING)
2     FROM EMP;
```

Error . . . not a single group set function

Como una alternativa, se puede colocar la función de grupo dentro del subquery y la columna regular en el query principal.

Ejemplo:

```
SQL> SELECT NOM_EMP, FEC_ING
2   FROM EMP
3   WHERE FEC_ING 0=
4   (SELECT MIN(FEC_ING) FROM EMP);

    NOM_EMP     FEC_ING
    JORGE       02-APR-90
```

III.7 PRIVILEGIOS

Seguridad de los datos

Se utiliza el comando GRANT de SQL para dar los siguientes privilegios:

- Privilegios del sistema.
- Privilegios sobre tablas.
- Privilegios sobre vistas.

Se utiliza el comando REVOKE para eliminar cualquier tipo de privilegios.

Privilegios del sistema

Existen 3 niveles de privilegios del sistema, que son los siguientes:

1) DBA: todos los privilegios.
2) RESOURCE: privilegio de conexión y creación de tablas.
3) CONNECT: privilegio de conexión unicamente consultas.

Unicamente un DBA tiene la autoridad para dar de alta nuevos usuarios.

A continuación se muestran algunos ejemplos con el comando GRANT.

```
SQL> GRANT CONNECT TO ADMIN
2   IDENTIFIED BY PEMEX;
```

Un usuario puede cambiar su propio password

```
SQL> GRANT CONNECT TO ADMIN
2   IDENTIFIED BY PEMEX1;
```

Privilegios sobre tablas

Existen 2 maneras de ganar acceso a tablas:

- Crear una tabla.
- Obtener permisos sobre tablas creadas por otros usuarios. Este lo debe otorgar el dueño de la tabla, de la siguiente manera:

```
SQL> GRANT SELECT, INSERT ON EMP
2       TO ADMIN;
```

Los siguientes privilegios pueden ser otorgados sobre tablas:

SELECT, INSERT, UPDATE, DELETE, ALTER E INDEX

Para dar a un usuario todos los privilegios sobre una tabla.

```
SQL> GRANT ALL ON EMP TO ADMIN;
```

Algo más acerca de privilegios sobre tablas

Cuando se accesa a la tabla de otro usuario, es posible que reciba el mensaje siguiente:

Table or view does not exist

Esto significa que:

- No se tiene acceso a dicha tabla o vista.
- La tabla o vista realmente no existe.

También podemos dar a otros el derecho de otorgar privilegios sobre nuestras tablas.

```
SQL> GRANT ALL
2    ON EMP
3    TO ADMIN
4    WITH GRANT OPTION;
```

Eliminar privilegios

Los privilegios del sistema solamente se eliminan con el DBA y los privilegios sobre tablas se pueden eliminar en cualquier momento.

Los privilegios de las tablas se eliminan de la siguiente manera:

```
SQL> REVOKE INSERT ON EMP
2    FROM ADMIN;
```

Sinónimos

Si otro usuario le ha dado privilegios de SELECT sobre alguna de sus tablas, se puede accesar de la siguiente manera:

```
SQL> SELECT * FROM  ADMIN.EMP;
```

donde:
ADMIN es el nombre del usuario dueño de la tabla.

Como una alternativa, se puede crear sinónimos utilizando un nombre más significativo y simple.

A continuación se muestra la forma de crear un sinónimo llamado EMPLEADOS sobre la tabla del usuario OPERA

```
SQL> CREATE SYNONYM EMPLEADOS
2     FOR OPERA.EMP;
```

Ahora se puede consultar la tabla EMPLEADOS como cualquier otra tabla.

III.8 INDICES

Indexando tablas

El propósito de los índices es ayudar al RDBMS a consultar tablas grandes de una forma más rápida. En lugar de leer toda la tabla, la recorre de una forma más eficiente.

Para crear un índice

```
SQL> CREATE INDEX EMP_IDX
2   ON EMP(NOM_EMP);
```

Para borrar un índice

```
SQL> DROP INDEX EMP_IDX;
```

Información sobre índices

- Indexar únicamente tablas grandes (por lo menos 50 renglones).
- Insertar datos antes de indexar.
- Una tabla puede tener varios índices.
- Generalmente se indexa la columna que identifica a los renglones en forma única (llave primaria).
- El uso de índices no tiene impacto sobre la sintaxis del SQL.
- Los índices son actualizados automáticamente.

Utilizando índices para evitar duplicidad de valores

Además de mejorar las consultas, se puede utilizar índices para asegurar que cada valor de una columna sea único.

Ejemplo:
asegurarse que cada número de empleado aparezca solamente una vez en al tabla EMP.

```
SQL> CREATE UNIQUE INDEX EMP_IDX
2   ON EMP(NUM_EMP);
```

Una vez creado el índice único, se obtendrá un mensaje de error si se desea insertar o actualizar un renglón con el mismo NUM_EMP que un renglón ya existente.

CAPITULO

SQL*PLUS Y APLICACIONES

IV

IV.1 COMANDOS DE SQL*PLUS

Además de los comandos estándar de SQL, el lenguaje SQL*Plus de ORACLE incluye comandos adicionales llamados comandos de SQL*Plus.

Estos comandos no se almacenan en el buffer de SQL.

Los comandos de SQL*Plus se pueden utilizar para generar reportes sofisticados, editar sentencias de SQL, proveer una facilidad de ayuda y mantener variables del sistema.

Los comandos de SQL*Plus son:

/	connect	help
append	del	input
break	describe	save
btitle	disconnect	spool
change	edit	show
column	exit	set
clear	get	ttitle

IV.2 REPORTES CON SQL*PLUS

El lenguaje SQL muestra la información tal y como está en la tabla, en contraste , SQL*Plus le permite cambiar los formatos de sus reportes.

En SQL*Plus no es necesario poner un punto y coma después de un comando.

Encabezados y pies de página

Para encabezados se utiliza el comando TTITLE.

Ejemplo:

SQL> TTITLE "SIMPLE REPORTE"

La fecha y número de página aparecen automáticamente hasta arriba de cada página cuando utilizamos TTITLE.

SQL> COLUMN NOM_EMP HEADING EMPLEADO

HEADING corresponde al encabezado de la columna y no necesita comillas si el encabezado es de una palabra.

Se puede cancelar la especificación de la columna limpiándola con el siguiente comando:

SQL> COLUMN NOM_EMP CLEAR

Formateo de columnas

Para formatear una columna se utiliza la cláusula FORMAT en el comando COLUMN.

Ejemplos:

SQL> COLUMN NOMBRE FORMAT A15

SQL > COLUMN SAL FORMAT $9,999.99

donde:

A15	indica que es un dato alfanúmerico de 15 caracteres.
$9,999.99	indica que es un dato numérico con dos decimales.

También dentro del comando COLUMN se emplea la cláusula Like que hace que una columna sea idéntica a otra.

Ejmplo:

SQL COLUMN COM LIKE SAL

Manejo de cortes

El comando BREAK se utiliza para suprimir la repetición de valores(cortes) y la cláusula ORDER BY es necesario para controlar los cortes.

Ejemplo:

```
SQL> BREAK ON NUM_DPTO
SQL> SELECT NUM_DPTO, NOM_EMP
2 FROM EMP
3 ORDER BY NUM_DPTO;

   NUM_DPTO   NOM_EMP
   20         PEDRO
              ALEX
              SAUL
              JORGE
```

Solamente un comando BREAK puede estar habilitado, pero se puede hacer cortes por más de una columna se muestra en el siguiente formato:

```
BREAK ON COL1 COL2 . . . COLn
```

Separando grupos de renglones

La cláusula SKIP se emplea par dejar espacios entre cortes.

Ejemplo:

```
SQL> BREAK ON NUM_DPTO SKIP 2
```

La cláusula SKIP en este caso deja un doble espacio entre cada departamento.

Con el comando CLEAR BREAK borra y deshabilita el comando BREAK.

Pueden hacerse cortes por página:

```
BREAK ON PAGE
```

Pueden hacerse cortes por reporte con:

```
BREAK ON REPORT
```

Pueden hacerse cortes con varias condiciones como:

BREAK ON PAGE ON REPORT

Desplegando cálculos por grupo

El comando COMPUTE se utiliza para hacer cálculos con las columnas.

Ejemplo:
SQL COMPUTE SUM OF SAL ON NUM_DPTO

El comando COMPUTE hace la suma de los salarios por cada departamento.

A continuación se muestran otros cálculos con el comando COMPUTE.

AVG	saca el promedio.
COUNT	número de valores no nulos.
MAX	valor máximo.
MIN	valor mínimo.
NUMBER	número de líneas.

El comando COMPUTE queda deshabilitado hasta que:

- Se deshabilita.
- Se reespecifica.
- Termina la sesión.

Para deshabilitarlo se debe da
CLEAR COMPUTE

Comandos de ambiente

Los parámetros de la sesión corriente de SQL*Plus se pueden desplegar tecleando:

SHOW opción
Ejemplo:
Mostrar el usuario con el que estamos conectado a SQL*Plus

SQL> SHOW USER

Para desplegar todos los parámetros se usa el comando SHOW ALL.

Para especificar valores de los parámetros se usa el comando SET.

SET opción valor

Ejemplo:
SQL> SET AUTOCOMMIT ON

Los comandos SET incluyen a los siguientes:

SET AUTOCOMMIT OFF/ON
ON habilita el commit automático.

SET ECHO OFF/ ON
ON los comandos ejecutados desde un archivo de comandos serán desplegados en la terminal.
OFF suprime el desplegado de los comandos.

SET FEEDBACK ON/OFF
ON los resultados de la consulta serán seguidos por un mensaje indicando el número de registros por la consulta.
OFF se suprimen los mensajes.

SET HEADING OFF/ON
ON se despliegan los encabezados de las columnas en los reportes.
OFF se suprimen los encabezados.

SET LINESIZE n
Fija el número de caracteres que SQL*Plus despliega por linea (el default es 80).

SET PAGESIZE n
Fija el número de líneas por página (el default es 24).

SET TERMOUT ON/OFF

ON permite recibir mensajes de otras terminales.

OFF no permite recibir mensajes de otras terminales.

Marcando valores nulos

El comando SET NULL se utiliza para marcar valores faltantes y el sustituto puede ser cualquier cadena.

Ejemplo.

```
SQL> SET NULL 'NO HAY DATOS'
SQL> SELECT NO_EMP, COM
2 FROM EMP
3 WHERE NUM_DPTO = 20;
```

NO_EMP	COM
PEDRO	200
ALEX	NO HAY DATOS
SAUL	NO HAY DATOS
JORGE	300

Salvando comandos a un archivo

Para salvar a disco un comando de SQL se utilza el comando SAVE como se muestra a continuación.

```
SAVE nombre_del_archivo
```

El sufijo .SQL es agregado por default al nombre del archivo(a menos que el nombre del archivo tenga un punto).

Ejemplo:

```
SQL> SELECT NO_EMP, NUM_EMP, PUESTO
2 FROM EMP
3 WHERE PUESTO = 'ANALISTA';
```

SQL> SAVE RESPALDO

Un archivo llamado RESPALDO.sql esta ahora en nuestro directorio.

Utilizando el procesador de textos del Sistema operativo

Se puede correr el editor del Sistema Operativo de SQL*Plus con el comando EDIT.
Ejemplo:
SQL> EDIT

Este comando edita el contenido del buffer y cualquier cambio hecho durante la edición se salva en el buffer.
Para regresar a SQL*Plus con salirse del editor bastará.

Para editar el contenido de un archivo, se le da al comando EDIT seguido por el nombre del archivo.
Ejemplo:
SQL> EDIT RESPALDO

*Comandos de edición de SQL*Plus*

Despliega el contenido del buffer de sql	LIST o L
Cambia la primera ocurrencia de cierto texto en una línea.	CHANGE o C
Agrega una línea al comando de SQL	INPUT o I
Agrega texto a una línea	APPEND o A
Borra una línea	DEL
Corre la instrucción que se encuentra en el buffer	

Recuperando comandos almacenados

El comando GET trae el contenido de un archivo al buffer de SQL y lo despliega en la pantalla.

Para recuperar el archivo RESPALDO dar el comando siguiente:

SQL> GET RESPALDO

El sufijo .SQL no se debe especificar.

Corriendo comandos almacenados

El comando START recupera el contenido de un archivo de comandos y lo corre.

Para recuperar y corre el archivo RESPALDO dar:

SQL> START RESPALDO

Un archivo de comandos ejecutado con el comando START puede contener comandos de SQL*Plus o más comandos de SQL.

Mandando la salida a un archivo

Para guardar los resultados de una consulta en un archivo y desplegar en la pantalla, utilizamos el comando SPOOL.

Si el nombre del archivo no tiene un punto, entonces se le agrega el sufijo .lst por default.

Para detener el SPOOL a un archivo, se le da el comando:

SQL> SPOOL OFF

Para imprimir los resultados de una consulta, se deben mandar a un archivo con el comando SPOOL nombre_archivo. Después en lugar de SPOOL OFF, se teclea:

SQL> SPOOL OUT

SPOOL OUT cierra el archivo de salida e imprime el archivo en al impresora de default del sistema.

Creando reportes en batch

A continuación se muestra un archivo que contiene comandos de SQL*Plus y comandos de SQL.

```
SET ECHO OFF
SET AUTOCOMMIT ON
SET PAGESIZE 25
INSERT INTO EMP(NO_EMO,NUM_EMP,FEC_ING)
VALUES('ELIZABETH',1000,SYSDATE);
INSERT INTO EMP(NO_EMP,NUM_EMP,NUM_DPTO)
VALUES('CESAR',1010,20);
SPOOL NUEVO_EMP
SELECT * FROM EMP
WHERE NUM_DPTO = 20
OR NUM_DPTO IS NULL
/
SPOOL OFF
SET AUTOCOMMIT OFF
```

ESTA
SALIR DE LA BIBLIOTECA

IV.3 APLICACIONES

Para mostrarles las apliciones de SQL a continuación se describe el siguiente sistema que se utiliza en Petróleos Mexicanos que opera en un ambiente de bases de datos de ORACLE.

SISTEMA DE COMPROBACION PRESUPUESTAL

Descripción general del sistema

El Sistema de Comprobación Presupuestal obtiene reportes de los gastos realizados por departamentos que permite que el departamento de sistemas y costos puedan validar su información con estos reportes.

El sistema obtiene la información de la tabla T_013, esta tabla pertenece al Sistema Institucional de Contabilidad (SIC) y además dicha información es validada con catálogos del Sistema Institucional del Control de Ejercicio Presupuestal (SICEP).

Para la obtención de los reportes se realizan los siguientes procesos:

1) Proceso de extracción : consiste en extraer la información de la tabla T_013 y pasarla a la tabla REGAS.

2)Proceso de filtrado y validación : consiste en dejar pasar la información que se encuentre en ciertos catalogos del SICEP.

3) Proceso de armado de registros : proceso por el cual se añaden a los datos adicionales de los catálogos del SICEP para efectuar la agrupación requerida para compararla contra el presupuesto.

Ver la figura IV.3.1.

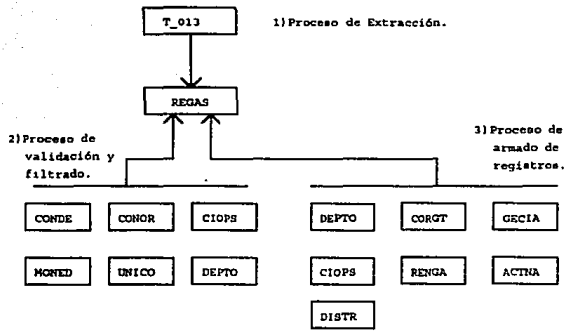


Figura IV.3.1 Procesos

Menu principal del sistema

El sistema está compuesto de un menu principal el cual consta de las siguientes opciones :

```
                    PETROLEOS MEXICANOS
S.L.                                          FECHA:16/02/94

          SISTEMA DE COMPROBACION PRESUPUESTAL

     1 Seleccion de la informacion

     2 Armado de registros

     3 Control de procesos

     4 Generacion de reportes

     5 Impresion de reportes

     6 Pantallas

     0 Salida del sistema

                              Seleccione una opcion y
                              Presione return ==>
```

A continuación se describen las funciones de las opciones del menú principal.

1. Selección de la información

Esta opción sirve para seleccionar el mes y año para la extracción de la información de la tabla T_013.

Al selecionar esta opción aparece la siguiente pantalla :

```
                    PETROLEOS MEXICANOS
S.L.                                        FECHA:16/02/94


   Teclee el mes a procesar(MM) >>
   Teclee el año que va a procesar(YY)>>
   Estan correctos el mes y el año (SI/NO):
   Desea realizar la seleccion y
   Filtrado de registros (SI/NO)>>
```

A continuación se describen las funciones de los programas que se utilizan en esta opción.

rengl.sql: Programa en SQL que borra la información de la tabla REGAS e inserta la información de la tabla T_013 a REGAS.

num_reg.sql: Programa en SQL que cuenta el número de registros obtenidos de la tabla T_013.

filreg.sql: Programa en SQL que le pone una letra A a todas las banderas de la tabla REGAS que se encuentran en catálogos de SICEP y además le pone otra letra A a la columna est_ado si todas las banderas tienen la letra A.

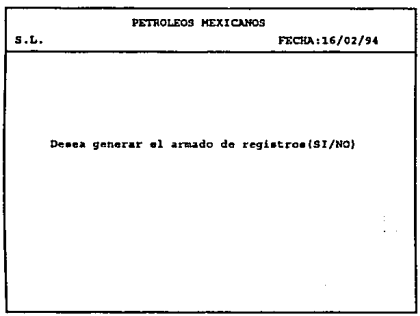
filcpsic.sql: Programa en SQL que muestra el número de registros con error encontrados durante el filtrado de la información.

filreg.pc: Programa en pro*c que selecciona todos los registros diferentes de A y los cambia por una letra según la bandera.

2. Armado de registros

Esta opción sirve para complementer y actualizar los registros.

Al seleccionar esta opción aparece una pantalla que se muestra en la figura siguiente:



A continuación se describen los programas que se utilizan en ésta opción.

arma.exe: Programa en shell que pregunta al usuario si desea realizar el armado de registros.

aseval.pc: Programa en pro*c que arma los registros de la tabla REGAS y valida la información con catálogos institucionales del SICEP.

regas2.pc: Programa en pro*c que sirve para adicionar el departamento económico correspondiente dependiendo del rango de la clave de autorización.

A continuación se describen las funciones de los programas

que se utilizan en ésta opción :

rengl.sql: Programa en SQL que se encarga de borrar la información de la tabla REGAS (tabla principal del sistema), selecciona la información de la tabla T_013 e inserta en la tabla REGAS.

num_reg.sql: Programa en SQL que sirve para contar el número de registros obtenidos de la tabla T_013.

filreg.sql: Programa en SQL que pone una letra A a todas las banderas de la tabla REGAS que se encuentra en catalogos institucionales del SICEP y después le pone otra letra A a la columna est_ado si todas las banderas tienen la letra A.

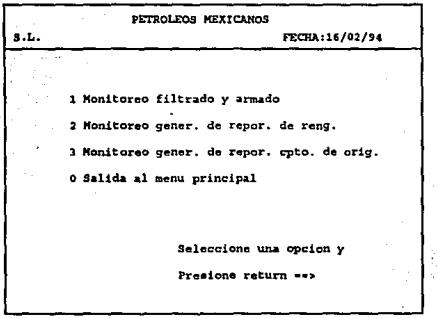
filcpsic.sql: Programa en SQL que muestra el número de registros con error encontrados durante el filtrado de la información.

filreg.pc: Programa en Pro*c que selecciona todos los registros diferentes de 'A' y cambia la columna por una letra.

3. Control de procesos

Esta opción se emplea para monitorear los procesos que se van realizando.

Al seleccionar esta opción aparece la pantalla siguiente:



Para la opción 1 el monitoreo es el siguiente :

Sistema Local de Costos
Carga de información de SIC para el sistema de Costos.
Inicia la carga de información a la tabla costos.
El número de registros obtenidos en la selección es de:
Inicia el filtrado de la tabla costos.
El número de registros con error encontrados es de:

Finaliza el filtrado de la información.

Inicia generación para registros con - sin proyecto
FECHA HORA
Inicia el armado de registros
Numero de registros leídos:
Numero de registros armados:
Numero de registros no armados:
Accesos a DEPTO
Accesos a RENGA
Accesos a GECIA
Accesos a CIOPS
Accesos a ACTNA
Accesos a CORGT

Finaliza la generación del armado por registro

FECHA HORA

Inicia el cambio de departamentos instit. a locales

FECHA HORA

Finaliza la actualización de departamentos

FECHA HORA

Fin del proceso.

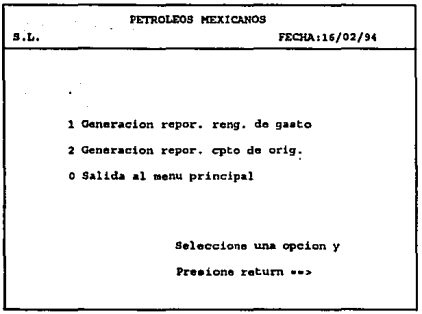
Para la opción 2 el monitoreo es el siguiente:

Gen. de reportes del pagado, recuperado y devengado
Proceso para la generación de reportes
Devengado gastos por renglón
Devengado sector por renglón
Devengado costos por renglón
Suma de las fases 2 y 3
Finaliza la generación de reportes.

4. Generación de reportes

En ésta opción se generan los reportes por renglón del Gasto y concepto de origen que se encuentran agrupados por fases, la 1era. fase (Devengado), 2a. fase (Ejercido) y 3a. fase (Flujo de efectivo).

Al seleccionar esta opción aparece el siguiente menu:



A continuación se muestran los programas, para generar los reportes por renglón de gasto:

costos.sql: Programa en SQL que se encarga de obtener reportes de la suma de las 3 fases.

costos1.sql: Programa en SQL que se encarga de obtener reportes primera fase.

costos2.sql: Programa en SQL que se encarga de obtener reportes de la segunda fase.

costos3.sql: Programa en SQL que se encarga de obtener reportes de la tercera fase.

gastos.sql: Programa an SQL que se encarga de obtener reportes de la suma de las tres fases.

gastos1.sql: Programa en SQL que se encarga de obtener reportes de la primera fase.

gastos2.sql: Programa en SQL que se encarga de obtener reportes de la segunda fase.

gastos3.sql: Programa en SQL que se encarga de obtener reportes de la tercera fase.

sptclar.sql: Programa de SQL que se encarga de obtener reportes de la suma de las tres fases.

sptciar1.sql: Programa de SQL que se encarga de obtener reportes de la primera fase.

sptciar2.sql: Programa en SQL que se encarga de obtener reportes de la segunda fase.

sptciar3.sql: Programa en SQL que se encarga de obtener reportes de la tercera fase.

A continuación se muestran los programas para obtener los reportes por concepto de origen.

con_spt1.sql: Programa en SQL que se encarga de obtener reportes de la primera fase.

con_spt2.sql: Programa en SQL que se encarga de obtener reportes de la segunda fase.

con_spt3.sql: Programa en SQL que se encarga de obtener reportes de la tercera fase.

sector1.sql: Programa que se encarga de obtener reportes de la primera fase.

sector2.sql: Programa que se encarga de obtener reportes de la segunda fase.

sector3.sql: Programa que se encarga de obtener reportes de la tecera fase.

En los reportes por renglón del gasto aparecen las siguientes columnas:

sect Es la clave del departamento donde se realiza el gasto.

renglon de gasto Es la clave de renglón del gasto afectado.

importe Es el importe contable o sea la cantidad gastada.

centro Es la clave delc entro de trabajo afectado.

En los reportes por concepto de origen aparecen las siguientes columnas:

SECT Es la clave del departamento donde se realiza el gasto.

CVE_MOMC Es la clave del momento contable.

CVE_ECOM	Es la clave del departamento económico.
NUM_CTAM	Es el número de cuenta de mayor.
CVE_CPTO	Es la clave del concepto de origen.
NDO_CON	Es el número de documento contable.
FEC_ELAB	Es la fecha de elaboración.
FEC_VENC	Es la fecha de vencimiento.
IMP_CONT	Es el importe contable.
CTA_CRB	Es la cuenta de mayor afectada por crédito.
T	Es el tipo de documento.

5. Impresión de reportes

Esta opción se utiliza para imprimir los reportes.
Al seleccionar ésta opción aparece el siguiente menú:

```
                    PETROLEOS MEXICANOS
S.L.                                          FECHA:16/02/94


      1 Imprime repor. de renglon del gasto
      2 Imprime repor. por concepto de origen
      0 Salida al menu principal


                      Seleccione una opcion y
                      Presione return ==>
```

Esta opción utiliza los siguientes programas:

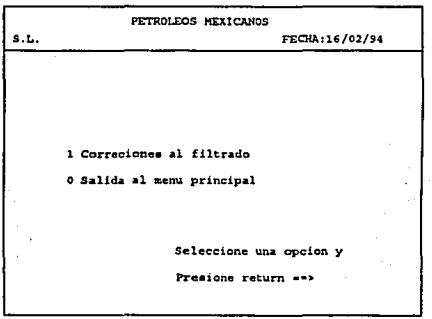
repor.sh Programa en shell que genera el menú de impresión de reportes.

imp_regas.sh Programa en shell para imprimir los reportes de renglón del gasto.

imp_comp.sh Programa en shell para imprimir los reportes por concepto de origen.

6.Pantallas

Esta pantalla se utiliza para corregir la información procesada.
Al selecionar ésta opción aparece el siguiente menú:



La forma siguiente esta desarrollada en la utileria de ORACLE llamada SQL*Forms que nos permite interactuar a través de ellas con la base de datos de ORACLE.

En esta opción la pantalla es la siguiente:

```
MODIF          PETROLEOS MEXICANOS          FECHA:14/02/94
        SISTEMA DE COMPROBACION PRESUPUESTAL C.T. 203

CONTADURIA _____          C.T. AFECTADOR _____
NO. DE DOCTO C. _____     MOMENTO CONTABLE _____
CONSECUTIVO _____         FOLIO DE ADEFAS _____
TIPO DE DOCTO C. _____
DOCTO. DE REF. _____
CUENTA DE CARGO _____     ESTADO DEL REG. _ _ _ _ _ _

C.T. AFECTADO _____       CLAVE DE MONEDA _____
DEPARTAMENTO _____        IMPORTE MON. NAL _____
CONCEPTO DE ORIGEN _____  IMPORTE MON. EXT _____
PROYECTO _____            TIPO DE CAMBIO _____
CLAVE DE AUTOR. _____     FECHA DE VENCIMIENTO _____
FECHA DE ELAB. _____      PRESIONE <RETORNO>
```

CONCLUSIONES

1. El sistema de base de datos relacional ofrece muchos beneficios como: fácil acceso a los datos, flexibilidad en el modelado de los datos, almacenamiento y reducción de datos redundantes.

2. Las bases de datos relacionales nos permite reducir al mínimo el mantenimiento a los programas debido a la independencia de los datos.

3. Otra ventaja que tienen las bases de datos relacionales sobre otras bases de datos, es la de poder manejar varios conjuntos de registros en vez de uno sólo.

4. El lenguaje SQL proporciona una extraordinaria flexibilidad y potencia para el manejo de datos.

5. El lenguaje SQL puede usarse dentro de programas o también directamente dentro de programas o también directamente desde una terminal, sin la necesidad de incluirlo previamente dentro de un programa. Esto lo hace útil para los programadores como para los usuarios finales que deseen la posibilidad de acceder a los datos mediante consultas abiertas no previamente programadas.

[redacted]

APENDICE

DESCRIPCION DE TABLAS

A

1) TABLA REGAS (TABLA PRINCIPAL DEL SISTEMA)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DEL CENTRO DE TRAB.	CVE_CTRO	NUMBER(3)
CLAVE DE MONEDA	CVE_MONE	CHAR(1)
NUMERO DE REGISTRO	NUM_REGI	NUMBER(5)
CLAVE DE DEPARTAMENTO	CVE_DPTO	NUMBER(5)
FECHA DE ELABORACION	FEC_ELAB	DATE
CLAVE DE CTA. DE AFECT.	CVE_CAFE	NUMBER(3)
CLAVE DE AUTORIZACION	CVE_AUTO	CHAR(11)
CLAVE DE CONCEPTO DE ORIG.	CVE_CPTO	NUMBER(6)
CLAVE DE UNIDAD DE CTRO. AF.	CVE_UCAF	NUMBER(3)
CLAVE DE GERENCIA	CEV_GCIA	NUMBER(5)
CLAVE DE CENTRO AFECTADO	CVE_CTAF	NUMBER(3)
CLAVE DE PROYECTO	CVE_PROY	CHAR(8)
CVE. DE RENGLON DEL GASTO	CVE_RENG	NUMBER(3)
IMPORTE DE MONEDA EXT.	IMP_MEXT	NUMBER(15,2)
CLAVE DE PROGRAMA	CVE_PROG	CHAR(2)
CLAVE DE SUBPROGRAMA	CVE_SUBP	NUMBER(2)
CVE. DE ACTIVIDAD/NAT.DEINV.	CVE_ACNA	NUMBER(2)
FOLIO DE ADEFAS	FOL_ADEF	CHAR(8)
CLAVE DE CONTADURIA	CVE_CONT	NUMBER(4)
NUMERO DE CTA. DE MAYOR	NUM_CTAM	NUMBER(4)
IMPORTE CONTABLE	IMP_CONT	NUMBER(15)
IDENTIFICACION DE ENVIO	IND_ENVI	CHAR(2)
ESTADO	EST_ADO	CHAR(1)
FECHA DE VENCIMIENTO	FEC_VENC	DATE
EQ. ENTRE M. EXT Y M. NAL.	SIC_TCAM	NUMBER(11,6)
DOCUMENTO DE REFERENCIA	DOC_REFE	CHAR(15)
FACTURA DE APLICACION	FAC_APLI	CHAR(2)
NUMERO DE DOCTO CONTABLE	NDO_CONT	CHAR(8)
TIPO DE DOCTO CONTABLE	TDO_CONT	CHAR(1)
BANDERA DE MAXI	BAN_MAXI	CHAR(1)
NUM. DE RENG. DOC. CONT.	R_SEC	NUMBER(5)
BANDERA DE CPTO. DE ORIG.	BAN_CO	CHAR(1)
BANDERA DE PROYECTO	BAN_PROY	CHAR(1)
BANDERA DE MONEDA	BAN_MONE	CHAR(1)
BANDERA DE CENTRO DE TRAB.	BAN_CTRO	CHAR(1)
BANDERA DE DEPARTAMENTO	BAN_DPTO	CHAR(1)
BANDERA DE IMPORTE	BAN_IM	CHAR(19
BANDERA DE INGRESO/EGRE	BAN_IE	CHAR(1)
BANDERA D CVE DE AUTORIZ.	BAN_AUTO	CHAR(1)
DIGITO DE AJUSTE	DIG_AJ	CHAR(1)
SIGNO	SIGNO	CHAR81)
CTA. DE MAYOR AF. POR CRED.	CTA_CRB	NUMBER(4)

2) TABLA T_013

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
NUMERO DE DOCUMENTO	N_DOC	NOT NULL CHAR(8)
TIPO DE DOCUMENTO	T_DOC	NOT NULL NUMBER(1)
NUM.DE RENG. DOC. CONT.	R_SEC	NOT NULL NUMBER(5)
NUMERO DE CONTADURIA	N_CONT	NOT NULL NUMBER(4)
NUMERO DE CTRO. AFECTADO	N_CENAFR	NOT NULL NUMBER(3)
NUMERO DE DEPENDENCIA	N_DEP	NOT NULL NUMBER(5)
FECHA DE ELABORACION	F_ELAB	NOT NULL DATE
NUM. DE CTRO. AFECTADO O.	N_CENAFRO	NOT NULL NUMBER(3)
CLAVE DE AUTORIZACION	CV_AUT	NOT NULL CHAR(11)
CLAVE DE PROYECTO	CV_PROY	CHAR(8)
FOLIO DE ADEFAS	FOL_ADEF	CHAR(8)
FECHA DE VENCIMIENTO	F_VENC	NOT NULL DATE
IMP. DE MONEDA EXTR.	I_ME	NUMBER(16,2)
TIPO DE MONEDA	T_MON	NOT NULL CHAR(1)
TIPO DE CAMBIO DE OPERACION	T_CAMOP	NUMBER(11,6)
NUM. CONCEPTO DE ORIGEN	N_CONORI	NOT NULL NUMBER(6)
IMPORTE DE OPERACION	I_OP	NOT NULL NUMBER(18,2)
DOCUMENTO FUENTE	DOC_FTE	CHAR(15)
DIGITO DE ADEFAS	DIG_ADE	CHAR(1)
CTA. DE DEBITO BANCARIO	CTA_DBB	NOT NULL NUMBER(4)
CTA. DE CREDITO BANCARIO	CTA_CRB	NOT NULL NUMBER(4)
CTA DE CREDITO ORDEN	CTA_CRO	NOT NULL NUMBER (4)
DIGITO ESPECIFICO	DIG_ESP	CHAR(1)
DIGITO DE AJUSTE	DIG_AJ	CHAR(1)
FECHA DE EXTRACCION A SICPI	F_SICPI	DATE

3) TABLA T_015 (DOCUMENTO PROGRAMADO IDC)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
NUMERO DE DOCUMENTO	N_DOC	NOT NULL CHAR(8)
TIPO DE DOCUMENTO	T_DOC	NOT NULL NUMBER(1)
FECHA DE VENCIMIENTO	F_VENC	NOT NULL DATE
SUMA DE LOS IMPOR.DOC. CONT.	I_TTDOC	NOT NULL NUMBER(18,2)
TIPO DE PAGO POR COBRAR	T_PAGCOB	CHAR(1)
IDEN. DE CTA. BANCARIA	ID_CTABAN	NUMBER(3)
NUMERO DE INP. DIARIO DE CAJA	N_IDC	NUMBER(3)
NUMERO DE POLIZA DE CAJA	N_POLCAJ	CHAR(9)
CLAVE DEL DOCUMENTO OP.	CV_OPD	CHAR(1)
NUMERO DEL PAGADOR	N_PAGR	NUMBER(2)
FECHA EN QUE LA CAJA PAGA	F_OP	DATE
NOMBRE DEL BENEFICIARIO	NOM_BEN	NOT NULL CHAR(30)
NUMERO DE CHEQUE	N_CHEQ	NUMBER(6)
IMPORTE DEL CHEQUE	I_CHEQ	NUMBER(18,2)

4) TABLA T_004 (DOCUMENTO CONTABILIZADOR)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
NUMERO DE DOCUMENTO	N_DOC	NOT NULL CHAR(8)
TIPO DE DOCUMENTO	T_DOC	NOT NULL NUMBER(1)
FECHA DE ELABORACION	F_ELAB	NOT NULL DATE
NUMERO DE CENTRO AFEC.	N_CENAFR	NOT NULL NUMBER(3)
AREA FUNCIONAL	AR_FUN	NOT NULL NUMBER(3)
TIPO DE MONEDA	T_MON	NOT NULL CHAR(1)
TIPO DE CAMBIO DE OP	T_CAMOP	NOT NULL NUMBER(11,6)
SUMA DE LOS IMPOR. DOC. CONT.	I_TTDOC	NOT NULL NUMBER(18,2)
SUMA DEL IMP MONEDA EXT.	I_TIME	NOT NULL NUMBER(16,2)
DESCRIPCION DEL CONCEPTO	DES-CON	NOT NULL CHAR(100)
NUMERO DE ERROR	N_ERROR	NUMBER(4)
CLAVE DE ATUALIZACION	CV_ACT	CHAR(1)
FECHA DE MOVIMIENTOS	F_MOV	NOT NULL DATE
NOMBRE DEL BENEFICIARIO	NOM_BEN	CHAR(30)

5) TABLA T_006 (DOCUMENTO CONTABLE)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
NUMERO DE DOCUMENTO	N_DOC	NOT NULL CHAR(8)
TIPO DE DOCUMENTO	T_DOC	NOT NULL NUMBER(1)
NUMERO DE RENG. DOC. CONT	R_SEC	NOT NULL NUMBER(1)
INGRESOS DE MONEDA EXT.	I_ME	NUMBER(16,2)
IMPORTE DE OPERACION	I_OP	NOT NULL NUMBER(18,2)
CLAVE DE OPERACION	CV_OP	CHAR(5)
CUENTA DE DEBITO	CTA_DBB	NOT NULL NUMBER(4)
SUBCUENTA DE DEBITO	SCTA_DB	NOT NULL CHAR(9)
CUENTA DE CREDITO DE ORD.	CTA_CRO	NOT NULL NUMBER(4)
CLAVE DE DEBITO O CREDITO	CV_DH	CHAR(1)
DIGITO ESPECIFICO	DIG_ESP	CHAR(1)
NUMERO DE AVISO	N_AVI	CHAR(15)
CLAVE DE VALIDACION	CV_VAL	CHAR(1)
CLAVE DE AVISOS TRANSM.	CV_AVITRA	CHAR(1)
DIGITO DE AJUSTE	DIG_AJ	CHAR(1)

6) TABLA T_007 (DOCUMENTO PRESUPUESTAL)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
NUMERO DE DOCUMENTO	N_DOC	NOT NULL CHAR(8)
TIPO DE DOCUMENTO	T_DOC	NOT NULL NUMBER(1)
NUMERO DE RENG. DOC. CONT	R_SEC	NOT NULL NUMBER(1)
DOCUMENTO FUENTE	DOC_FTE	NOT NULL CHAR(1)
FOLIO DE ADEFAS	FOL_ADEF	CHAR(8)
CLAVE DE PROYECTO	CVE_PROY	CHAR(8)
CLAVE DE AUTORIZACION	CVE_AUTO	NOT NULL CHAR(11)
FECHA DE VENCIMIENTO	F_VENC	NOT NULL DATE
NUMERO DE CENTRO AFEC.	N_CENAFO	NOT NULL NUMBER(3)
NUMERO DE DEPENDENCIA	N_DEP	NOT NULL NUMBER(5)
NUMERO DE CONCEPTO DE OR.	N_CONORI	NOT NULL NUMBER(6)
DIGITO DE ADEFAS	DIG_ADE	CHAR(1)

7) TABLA DEPTO (CATALOGO DE DEPARTAMENTOS)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DEL CENTRO	CVE_CTRO	NOT NULL NUMBER(3)
CLAVE DEL DEPARTAMENTO	CVE_DPTO	NOT NULL NUMBER(5)
CVE. DE CENTRO DE TRAB. RENG	CVE_CTRG	NOT NULL NUMBER(3)
CLAVE DE LA GERENCIA	CVE_GCIA	NOT NULL NUMBER(5)
CVE. UNIDAD DE CONTROL	CVE_UDCO	NOT NULL NUMBER(3)
NUMERO DE CTA. DE MAYOR	NUM_CTAM	NOT NULL NUMBER(4)
NUMERO DE CTA DE OP.	NUM_CTAO	NOT NULL NUMBER(6)
DESCRIP. DEL DEPARTAMENTO	DES_DPTO	NOT NULL CHAR(50)
CLAVE DEL NIVEL	CVE_NIVL	NOT NULL NUMBER(2)
TIPO DE REGRISTRO	TIP_REGI	NOT NULL CHAR(1)
CVE. DE OFICINA PAGADORA	CVE_OFIP	NOT NULL NUMBER(3)

8) TABLA CIOPS (CATALOGO DE OBRAS Y PROYECTOS)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DEL PROYECTO	CVE_PROY	NOT NULL CHAR(8)
CLAVE DE PARTIDA	CVE_PART	NOT NULL CHAR(8)
DESCRIP. DEL PROYECTO	DES_PROY	NOT NULL CHAR(40)
LOCALIZACION DE LA OBRA	LOC_OBRA	NOT NULL CHAR(50)
VIGENCIA	VIGENCIA	NOT NULL NUMBER(3)
TIPO DE PRESUP. AUTORIZ.	TIP_PPTO	NOT NULL NUMBER(3)
CVE. CENTRO DE TRAB. RENG.	CVE_CTRG	NOT NULL NUMBER(3)
CLAVE DE LA GERENCIA	CVE_GCIA	NOT NULL NUMBER(5)
NUMERO DE PUESTO BASE	NUM_PBAS	NOT NULL CHAR(8)
TIPO DE REGISTRO	TIP_REGI	NOT NULL CHAR(1)

9) TABLA CONDE (CATALOGO DE RELACIONES C. O. - DEPTO)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DEL DEPARTAMENTO	CVE_DPTO	NOT NULL NUMBER(5)
CLAVE DE CONCEPTO DE ORG.	CVE_PART	NOT NULL NUMBER(6)
TIPO DE REGISTRO	TIP_REGI	NOT NULL CHAR(1)

10) TABLA CONOR (CATALOGO DE CONCEPTO DE ORIGEN)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DE CONCEPTO DE ORG.	CVE_CPTO	NOT NULL CHAR(6)
DESCRIPCION DE C. O.	DES_CPTO	NOT NULL CHAR(55)
CODIGO CARACTERISTICO	COD_CARA	NOT NULL NUMBER(6)
TIPO DE REGISTRO	TIP_REGI	NOT NULL CHAR(1)
IDENTIFICACION DE PROY.	IND_PROY	NOT NULL CHAR(1)

11) TABLA UNICO (CATALOGO DE UNIDAD DE CONTROL)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DE UDCO	CVE_UDCO	NOT NULL NUMBER(3)
DESCRIPCION DE UDCO	CVE_UDCO	NOT NULL CHAR(40)
TIPO DE REGISTRO	TIP_REGI	NOT NULL CHAR(1)

12) TABLA MONEDCONOR (CATALOGO DE MONEDAS)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DE MONEDA	CVE_MONE	NOT NULL CHAR(1)
DESCRIPCION DE C. O.	DES_CPTO	NOT NULL CHAR(1)
TIPO DE REGISTRO	TIP_REGI	NOT NULL CHAR(1)

13) TABLA CETRO (CATALOGO DE CENTROS DE TRABAJO)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DE CENTRO DE TRAB.	CVE_CTRO	NUMBER(3)
CLAVE DE DEPARTAMENTO	CVE_DPTO	NUMBER(5)
CLAVE DE ECONOMICO	CVE_ECOM	NUMBER(5)
CLAVE DE RENGLON DEL GASTO	CVE_RENG	NUMBER(3)
CLAVE DEL PROGRAMA	CVE_PROG	CHAR(2)
INGRESOS-EGRESOS	ING_EGR	CHAR(1)
IMPORTE PRESUP	IMP_PRE	NUMBER(2)
CLAVE DEL AÑO	CVE-ANIO	NUMBER(2)
CLAVE DEL MES	CVE_MES	NUMBER(2)

14) TABLA RENGA (CATALOGO DE RENGLONES DEL GASTO)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DEL RENGLON DEL G.	CVE_RENG	NOT NULL NUMBER(3)
DESCRIPCION DEL R. G. .	DES_RENG	NOT NULL CHAR(50)
TIPO DE REGISTRO	TIP_REGI	NOT NULL CHAR(1)
CLAVE DEL GRUPO	CVE_GPOF	NUMBER(3)
CLAVE DEL PROGRAMA	CVE_PROG	CHAR(2)
CLAVE DE SUPTCIA.	CVE_SPGR	NUMBER(2)
CLAVE DE INGRESOS/EGRE.	CVE_EGIN	NUMBER(2)

15) TABLA T_041 (AMBIENTE DE BASE DE DATOS)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
NUMERO DE CONTADURIA	N_CONT	NOT NULL NUMBER(4)
NUMERO DE CENTRO AFECT.	N_CENAFR	NOT NULL NUMBER(3)

16) TABLA ACTNA (CATALOGO DE ACTIVIDAD NAT-INVER.)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DEL PROGRAMA	CVE_PROG	NOT NULL CHAR(2)
CVE DE SUBGCIA ACT.	CVE_SUBA	NOT NULL NUMBER(2),
CVE. DE ACTIVIDAD - NAT.	CVE_NATA	NOT NULL NUMBER(2)
CLAVE DE CENTROS DE TRAB.	CVE_CTRG	NOT NULL NUMBER(3)
CLAVE DE GCIA.	CVE_GCIA	NUMBER(5)
DESC. NATURALEZA INVER.	DES_NATI	NOT NULL CHAR(40)
TIPO DE REGISTRO	TIP_REGI	NOT NULL CHAR(1)

17) TABLA CORGT (CATALOGO DE RELACION DE C. O. - R. G.)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DE RENG. DEL GASTO	CVE_RENG	NOT NULL NUMBER(3)
CLAVE DE RELACION C.O - R. G.	CVE_CPTG	NOT NULL CHAR(6)
TIPO DE REGISTRO	TIP_REGI	NOT NULL CHAR(1)

18) TABLA GCIA (CATALOGO DE GERENCIAS)

DESC. DEL CAMPO	MNEMONICO	TIPO DE DATO
CLAVE DEL CENTRO DE TRAB.	CVE_CTRO	NUMBER(3)
CVE DE LA GERENCIA	CVE_GCIA	NOT NULL NUMBER(5)
CVE. DE CENTROS DE TRABAJO	CVE_CTRS	NOT NULL NUMBER(3)
CLAVE DE COORDINACION	CVE_COOR	NOT NULL NUMBER(5)
DESCRIPCION DE LA GCIA.	DES_GCIA	NOT NULL CHAR(50)
TIPO DE REGISTRO	TIP_REGI	NOT NULL CHAR(1)

APENDICE

EJEMPLOS DE PROGRAMAS

B

```
REM ************************************************
REM *    PROGRAMA: gastos1.sql                      *
REM ************************************************
set termout off
set feedback off
set pagesize 60
set linesize 80
set underline =
column IMPORTE format 999,999,999,999,999;
ttitle center "TELECOMUNICACIONES VENTA DE CARPIO 1era. FASE" -
skip 2 -
center "GASTOS1" -
right "Fecha:  " fecha -
skip 1 -
left "Programa : gastos1" -
right "Reporte : TELEVC1" -
skip 2 -
left"*************************************************************"
btitle skip 1 format 999 center "pag." sql.pno skip "^L"
column today noprint new_val fecha;
break on SECTOR  skip 1 on report;
compute sum of IMPORTE on SECTOR
compute sum of IMPORTE on report
spool  televc1
select to_char(sysdate,'DD/MM/YY') today,
substr(cve_ecom,1,4) SECTOR, cve_reng "RENGLON DEL GASTO"
, sum(imp_cont) IMPORTE, cve_cetro centro
from regas
where tdo_cont in (3)
and cve_momc in (14)
and num_ctam in (1507,1508,2404,5219,5223,5221)
and cve_reng not in (237,232)
and cta_crb in (2102,2113,2118)
and fol_adef is null
and cve_ecom = 85000
group by substr(cve_ecom,1,4), cve_reng, cve_cetro
order by substr(cve_ecom,1,4)
/
```

```
REM ***********************************************
REM *    PROGRAMA: costos.sql                     *
REM ***********************************************
set termout off
set feedback off
set pagesize 60
set linesize 80
set underline -
column IMPORTE format 999,999,999,999,999;
ttitle center "DEPTOS. DE LA TECNICA ADMINISTRATIVA" -
skip 2 -
center "REPORTE GLOBAL" -
right "Fecha:  " fecha -
skip 1 -
left "Programa : COSTOS" -
right "Reporte : DPTADM" -
skip 2 -
left"-------------------------------------------------------------"
btitle skip 1 format 999 center "pag." sql.pno skip "^L"
column today noprint new_val fecha;
break on SECTOR  skip 1 on report;
compute sum of IMPORTE on SECTOR
compute sum of IMPORTE on report
spool  dptadm
select to_char(sysdate,'DD/MM/YY') today,
substr(cve_ecom,1,4) SECTOR, cve_reng "RENGLON DEL GASTO"
, sum(imp_cont) IMPORTE, cve_cetro centro
from regas
where tdo_cont in (2,3,4)
and cve_momc in (14,15,16)
and num_ctam in (1507,1508,2404,5219)
and cve_reng not in (237,232)
and cta_crb in (2110,1102,2102,2113,2118)
and fol_adef is null
and cve_ecom = 85002
group by substr(cve_ecom,1,4), cve_reng, cve_cetro
order by substr(cve_ecom,1,4)
/
```

```
REM ***********************************************
REM *       PROGRAMA: sptciar.sql                 *
REM ***********************************************
set termout off
set feedback off
set pagesize 60
set linesize 80
set underline =
column IMPORTE format 999,999,999,999,999;
ttitle center "ALMACEN VENTA DE CARPIO" -
skip 2 -
center "AGRUPACION POR SECTOR" -
right "Fecha:  " fecha -
skip 1 -
left "Programa : SPTCIAR" -
right "Reporte : ALMAVC" -
skip 2 -
left"============================================================"
btitle skip 1 format 999 center "pag." sql.pno skip "^L"
column today noprint new_val fecha;
break on DEPARTAMENTO skip 1 on report;
compute sum of IMPORTE on DEPARTAMENTO
compute sum of IMPORTE on report
spool  almavc
select to_char(sysdate,'DD/MM/YY') today,
cve_ecom DEPARTAMENTO, cve_reng "RENGLON DEL GASTO"
, sum(imp_cont) IMPORTE, cve_cetro centro
from regas
where tdo_cont in (2,3,4)
and cve_momc in (14,15,16)
and num_ctam in (1507,1508,2404,5219,5223,5221,5218,5214)
and cve_reng not in (237,232)
and cta_crb in (1102,2102,2110,2113,2118)
and fol_adef is null
and substr(cve_ecom,1,2)||substr(cve_ecom,4,5) = 4520
group by cve_ecom, cve_reng, cve_cetro
order by cve_ecom
/
```

```
REM ************************************************
REM *     PROGRAMA: con_spt1.sql                     *
REM ************************************************
set termout off
set feedback off
set pagesize 60
set linesize 125
set underline =
column IMPORTE format 999,999,999,999,999;
ttitle center "SPTCIA. DE EVALUC. SERVC. SIST INFOR. ALMAC. TESOR. ING.
TEL. ALMACEN 1era. FASE" -
skip 2 -
center "AGRUPACION POR SECTOR" -
right "Fecha:  " fecha -
skip 1 -
left "Programa : SPTCIAR" -
right "Reporte : EVALUC1" -
skip 2 -
left"============================================================"
btitle skip 1 format 999 center "pag." sql.pno skip "^L"
column today noprint new_val fecha;
break on cve_momc on sector skip 2 on report;
compute sum of imp_cont on cve_momc
compute sum of imp_cont on sector
compute sum of imp_cont on num_ctam
spool  evaluc1
select to_char(sysdate,'DD/MM/YY') today,
substr(cve_ecom,1,4)  SECTOR,  cve_momc,  cve_ecom,  num_ctam,  cve_cpto,
ndo_cont,  fec_elab,fol_adef,fec_venc, imp_cont, cta_crb
from regas
where tdo_cont in (3)
and cve_momc in (14)
and num_ctam in (1507,1508,2404,5219,5223,5221,5218,5214)
and cve_reng not in (237,232)
and cta_crb in (2102,2110,2113,2118)
and fol_adef is null
and substr(cve_ecom>= 48000 and cve_ecom <= 90600
order by cve_momc,cve_ecom,num_ctam,cve_cpto
/
```

```
REM ***********************************************
REM *       PROGRAMA: sector3.sql                 *
REM ***********************************************
set termout off
set feedback off
set pagesize 62
set linesize 128
set underline =
column IMPORTE format 999,999,999,999,999;
ttitle center "SPTCIA. DE EVALUC: SERV: SIST INFOR: ALMAC. TESOR. ING.
TEL. 3era. FASE" -
skip 2 -
center "AGRUPACION POR SECTOR" -
right "Fecha:  " fecha -
skip 1 -
left "Programa : SPTCIAR" -
right "Reporte : EVALUC3" -
skip 2 -
left"=============================================================="
btitle skip 1 format 999 center "pag." sql.pno skip "^L"
column today noprint new_val fecha;
break on cve_momc on sector skip 2 on report;
compute sum of IMPORTE on SECTOR
compute sum of IMPORTE on report
compute sum of IMPORTE on num_ctam
spool  evaluc1
select to_char(sysdate,'DD/MM/YY') today,
substr(cve_ecom,1,4)  SECTOR,  cve_momc,  cve_dpto,  num_ctam,  cve_cpto,
ndo_cont,   fec_elab,fol_adef,fec_venc,   imp_cont,   cta_crb,   did_aj,
tdo_cont
from regas
where tdo_cont in (2,3,4)
and cve_momc in (14,15,16)
and num_ctam in (1507,1508,1606,2404,5219)
and cve_reng not in (237,232)
and cta_crb in (1102)
and fol_adef is null
and substr(cve_ecom>= 48000 and cve_ecom <= 90600)
order by cve_momc,cve_ecom,num_ctam,cve_cpto
/
```

```
REM ************************************************************
REM *      PROGRAMA: rengl.sql                                 *
REM ************************************************************
delete regas
/
insert into regas (cve_ctro,cve_dpto,doc_refe,cve_auto
,cve_momc,cve_cafe,cve_proy,cve_cpto,est_ado,imp_cont,r_sec
fec_elab,cev_mone,imp_mext,fol_adef,cve_cont,sic_tcam,num_ctam,
ban_co,ban_proy,ban_mon,ban_ctro,ban_dpto,
ban_im,ban_ie,ban_auto,dig_aj,fec_venc,tdo_cont,ndo_cont,cta_crb)
select n_cenafo,n_dep,doc_fte,cv_auto,substr(cta_cro,3,2),
n_cenafr,cv_proy,n_conori,'D',i_op,r_sec,,f_elab,t_mon,i_me,
fol-ade,n_cont,t_camop,cta_dbb,'D','Y','E','A','M','P','Q',
dig_aj,f-venc,t-doc,n_doc,cta_crb
from t_013
where t-doc in (3,4) and to_char(to_date(f_elab),'MMYY')|&1
and cta_cro in (7316,7317,7314=
/
insert into regas (cve_ctro,cve_dpto,doc_refe,cve_auto
,cve_momc,cve_cafe,cve_proy,cve_cpto,est_ado,imp_cont,r_sec
fec_elab,cev_mone,imp_mext,fol_adef,cve_cont,sic_tcam,num_ctam,
ban_co,ban_proy,ban_mon,ban_ctro,ban_dpto,
ban_im,ban_ie,ban_auto,dig_aj,fec_venc,tdo_cont,ndo_cont,cta_crb)
select n_cenafo,n_dep,doc_fte,cv_auto,substr(cta_cro,3,2),
n_cenafr,cv_proy,n_conori,'D',i_op,r_sec,,f_elab,t_mon,i_me,
fol-ade,n_cont,t_camop,cta_dbb,'D','Y','E','A','M','P','Q',
dig_aj,t_013.f_venc,t_013.t_doc,t_013.n_doc,cta_crb
from t_013,t_15
where t_013.n_doc = t_013.n_doc and t_013.t_doc in (1,2)
and to_char(to_date(f_op),'MMYY')=&1
and cta_cro in (7315,7325,7326)
/
insert into regas (cve_ctro,cve_dpto,doc_refe,cve_auto
,cve_momc,cve_cafe,cve_proy,cve_cpto,est_ado,imp_cont,r_sec
fec_elab,cev_mone,imp_mext,fol_adef,cve_cont,sic_tcam,num_ctam,
ban_co,ban_proy,ban_mon,ban_ctro,ban_dpto,
ban_im,ban_ie,ban_auto,dig_aj,fec_venc,tdo_cont,ndo_cont,cta_crb)
select n_cenafo,n_dep,doc_fte,cv_auto,substr(cta_cro,3,2),
n_cenafr,cv_proy,n_conori,'D',i_op,r_sec,,f_elab,t_mon,i_me,
fol-ade,n_cont,t_camop,cta_dbb,'D','Y','E','A','M','P','Q',
dig_aj,t_007.f_venc,t_007.t_doc,t_007.n_doc,cta_crb
from t_004,t_006,t_007
where to_char(to_date(t_004.fec_elab),'MMYY') =&1
and t_004.n_doc = t_006.n_doc and t_004.n_doc = t_007.n_doc
and t_004.t_doc = t_006.t_doc and t_004.t_doc = t_007.t_doc
and t_006.r_sec = t_007.r_sec
and t_007.n_conori in (928700,929200,929300)
and (cta_dbb = 1611 or cta_crb = 1611)
/
insert into regas (cve_ctro,cve_dpto,doc_refe,cve_auto
,cve_momc,cve_cafe,cve_proy,cve_cpto,est_ado,imp_cont,r_sec
fec_elab,cev_mone,imp_mext,fol_adef,cve_cont,sic_tcam,num_ctam,
ban_co,ban_proy,ban_mon,ban_ctro,ban_dpto,
```

```
ban_im,ban_ie,ban_auto,dig_aj,fec_venc,tdo_cont,ndo_cont,cta_crb)
select n_cenafo,n_dep,doc_fte,cv_auto,substr(cta_cro,3,2),
n_cenafr,cv_proy,n_conori,'D',i_op,r_sec,,f_elab,t_mon,i_me,
fol-ade,n_cont,t_camop,cta_dbb,'D','Y','E','A','M','P','Q',
dig_aj,t_007.f_venc,t_007.t_doc,t_007.n_doc,cta_crb
from t_004,t_006,t_007
where to_char(to_date(t_004.fec_elab),'MMYY') =&1
and t_004.n_doc = t_006.n_doc and t_004.n_doc = t_007.n_doc
and t_004.t_doc = t_006.t_doc and t_004.t_doc = t_007.t_doc
and t_006.r_sec = t_007.r_sec
and scta_cr in ('201869200',207879200')
and cta_cro in (7334,7314)
/
commit;
```

```
REM ************************************************
REM *     PROGRAMA: filreg.sql                       *
REM ************************************************
spool err_18.lis
update regas
set ban_proy = 'A'
/
update regas
set ban_co = 'A'
where cve_cpto in (select cve_cpto from conor)
/
update regas
set ban_mon = 'A'
where cve_mone in(select cve_mone from moned)
or cve_mone is null or cve_mone = ' '
/
update regas
set ban_ctro ='A'
where cve_dpto || cve_ctro in (select cve_dpto || cve_ctro from depto)
/
update regas
set ban_dpto = 'H'
where cve_cpto in (select cve_cpto from conde)
/
update regas
set ban_dpto = 'A'
where cve_dpto || cve_cpto in (select cve_dpto || abs(cve_cpto) from
conde)
and ban_dpto = 'H'
/
update regas
set  ban_im = 'A'
where imp_cont > 0 and (cve_mone = ' '  or cve_mone is null or cve_mone
= 'N')
/
update regas
set ban_ie = 'A'
where imp_ext > 0 and (cve_mone = ' ' or cve_mone is not null)
or (imp_mext is null or imp_mext = 0) and (cve_mone is null or cve_mone
= ' '
or cve_mone = 'N')
/
update regas
set ban_auto = 'A'
where substr(cve_auto,5,3) in (select cve_udco from unico)
and substr(cve_auto,2,2) <= (select to_char(last_day(to_date(decode
and substr(cve_auto,2,2),
'A','01','B','02','C','03','D','04','E','05','F',
'06','G','07','H','08','J','09','K','10','L','11','M','12') || '9'
|| substr(cve_auto,4,1), 'mmyy')),'dd') from dual)
/
update regas
set est_ado = 'A'
where ban_proy = 'A' and ban_co ='A' and ban_mon = 'A'
```

```
and ban_ctro = 'A' and ban_dpto = 'A' and ban_im = 'A'
and ban_ie = 'A' and ban_auto = 'A'
/
commit;
spool off
```

```
REM ********************************************************
REM *      PROGRAMA: num_reg.sql                           *
REM ********************************************************
ttitle "El numero de registros obtenidos en la seleccion es de:" suma
column count(*) noprint new_val suma
select count(*) from regas;
```

```
REM ********************************************************
REM *      PROGRAMA: FILCPSIC.sql                          *
REM ********************************************************
ttitle "El numero de registros con error encontrados es de:" suma
column count(*) noprint new_val suma
select count(*) from regas
where est_ado != 'A';
```

[redacted]

APENDICE

EJEMPLOS DE REPORTES

C

STCIA. DE EVALUC. SERVC. SIST INFOR ALMAC. TESOR. ING. TEL. 3era. FASE

Programa : SPTCIAR

AGRUPACION POR SECTOR

Fecha : 08/09/93
Reporte : EVALUC3

SECT	CVE_MONC	CVE_OPTO	NUM_CTAM	CVE_CPTO	MOD_CONT	FEC_ELAB	FOL_ADEF	FEC_VENC	IMP_CONT	CTA_CRB	D	T
7300	16	73000	5219	531101	93M17931	12-AUG-93		12-AUG-93	140	1102	4	
		73000		531101	93M17989	06-AUG-93		06-AUG-93	200	1102	4	
			sum						340			
sum									340			
7500		75000	5219	401500	93M17148	16-AUG-93		16-AUG-93	135	1102	4	
			sum						135			
sum									135			
sum									475			

Programa : SPTCIAR

AGRUPACION POR SECTOR

Fecha : 08/09/93
Reporte . EVALUC1

SECT	CVE_MOMC	CVE_ECOM	NUM_CTAM	CVE_CPTO	MOD_CONT	FEC_ELAB	FOL_ADEF	FEC_VENC	IMP_CONT	CTA_CRB	D	T
4810	14	48100	5217	0100	93M10550	26-AUG-93		24-SEP-93	3809.19	2102	3	
		**********	sub						3809.19			
****									----------			
sub									3809.19			
7671		75710	5223	311005	93M10535	20-AUG-93		10-SEP-93	16359.54	2110	3	
		75710		542000	93M10541	20-AUG-93		10-SEP-93	1047.6	2110	3	
		**********	sub						17407.14			
****									----------			
sub									17407.14			
	**********								----------			
	sub								21216.33			

TELEVISORES [illegible] VENTA DE CAMPO Tipo PASE

Programa COSIO1 GASTOS1 Fecha Reporte 16/07/93 TELEVC1

SECT	RENGLON DEL GASTO	IMPORTE	CENTRO
8500	300	7,170	203
	200	46,775	203
****			
SUB		53,945	
		53,945	

DEPTOS DE LA TECNICA ADMINISTRATIVA

Programa : COSTOS

REPORTE GLOBAL

Fecha 08/09/91-
Reporte OPTADM

RENGLON DEL GASTO	IMPORTE	CENTRO
234	340	203
	340	
222	135	800
	135	
	475	

ALMACEN VENTA DE CARPID

AGRUPACION POR SECTOR

Programa : BPTCIAR

Fecha : 08/09/93
Reporte : ALMAVC

DEPARTAMENTO	RENGLON DEL GASTO	IMPORTE	CENTRO
45020	202	11,456	283
SUB		11,456	
		11,456	

BIBLIOGRAFIA

1. JAMES MARTIN.
   ORGANIZACION DE LAS BASES DE DATOS
   PRENTICE HALL
   MEXICO 1989.

2. ULLMAN
   PRINCIPLES OF DATABASE SYSTEM
   COMPUTER SCIENCE PRESS
   E.U.A. 1984.

3. J. M. OLAIZOLA BARTOLOME
   SQL PARA USUARIOS Y PROGRAMADORES
   PARANINFO
   MEXICO 1990

4. GEORGE KOCH
   ORACLE VERSION 5 Y 6
   MC. GRAW HILL
   E.U.A. 1987

5. SQL LANGUAGE REFERENCE MANUAL
   VERSION 6.0
   ORACLE
   E.U.A. 1987.

6. SQL*PLUS REFERENCE GUIDE
   VERSION 2.0
   ORACLE
   E.U.A. 1987

7. ORACLE RDBMS DATABASE ADMINISTRATOR'S GUIDE
   VERSION 6.0
   ORACLE
   E.U.A. 1990.

8. HENRY F. KORTH
   FUNDAMENTOS DE BASES DE DATOS
   MC. GRAW HILL
   MEXICO 1987.

9. KRUGLINSKI DAVID
SISTEMAS DE ADMINISTRACION DE BASES DE DATOS
MC. GRAW HILL
MEXICO 1987.