Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

ACTUALIZACIÓN A DICIEMBRE 2009, DEL DATAMART DE INFORMACIÓN ACADÉMICA DE LOS ESTUDIANTES DE LA FACTULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Julio René Gatica Guzmán
Erick Orlando Villagrán Zelada
Asesorado por el Ing. Jorge Armín Mazariegos
Co-asesora Inga. Gladys Sucely Aceituno

Guatemala, octubre de 2011

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÌA

ACTUALIZACIÓN A DICIEMBRE 2009, DEL DATAMART DE INFORMACIÓN ACADÉMICA DE LOS ESTUDIANTES DE LA FACTULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JULIO RENÉ GATICA GUZMÁN
ERICK ORLANDO VILLAGRÁN ZELADA
ASESORADO POR EL ING. JORGE ARMIN MAZARIEGOS
CO-ASESORADO POR LA INGA. GLADYS SUCELY ACEITUNO

AL CONFERÍRSELES EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, OCTUBRE DE 2011

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Sydney Alexander Samuels Milson
EXAMINADORA	Inga. Claudia Licet Rojas Morales
EXAMINADOR	Ing. Cesar Fernández Cáceres
EXAMINADOR	Ing. Manuel Fernando López Fernández
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración nuestro trabajo de graduación titulado:

ACTUALIZACIÓN A DICIEMBRE 2009, DEL DATAMART DE INFORMACIÓN ACADÉMICA DE LOS ESTUDIANTES DE LA FACTULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Tema que nos fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha abril de 2008.

Julio René Gatica Guzmán

Erick Orlando Villagrán Zelada

AGRADECIMIENTOS A:

Mi madre	Rosa Guzmán por su apoyo incondicional, ya que sin su ayuda no lo hubiera logrado.
Mis hermanos	Héctor, María y Mario por su ayuda y apoyo.
Mis sobrinos y demás familia	Por estar siempre pendientes.
Mis amigos	Qué en los momentos difíciles siempre estaban dispuestos a ayudar.

Julio René Gatica Guzmán

ACTO QUE DEDICO A:

Mi madre	Rosa Guzmán por su esfuerzo, trabajo, dedicación, para sacarnos a mis hermanos y a mí adelante.
Mis hermanos	Héctor, María y Mario por creer en mí y darme su apoyo en toda la carrera.
Mis sobrinos y demás familia	Por su cariño y apoyo.
Mis amigos	Por ser apoyo en tiempos difíciles, estar presentes en las buenas y malas.

Julio René Gatica Guzmán

AGRADECIMIENTOS A:

Dios	Por su infinito amor y darme la oportunidad de concluir esta meta con éxito.
Mis padres y hermana	Qué con su cariño y confianza me han apoyado en todo momento.
Mi esposa	Por su amor y en especial por el especial apoyo en la culminación de este trabajo.
Mi familia	Por su ayuda y apoyo en todo momento.
Mis amigos y sus familias	Por darnos ese apoyo desinteresadamente.
Universidad de San Carlos de Guatemala	Por darnos la oportunidad de superarnos cada día más, en cada una de sus aulas.
Centro de Cálculo de Ingeniería	Por apoyarnos en la realización de este trabajo.
Usted	Especialmente, gracias por ser partícipe de este triunfo.

Erick Orlando Villagrán Zelada

ACTO QUE DEDICO A:

Dios	Que con ÉL todo es posible.
Mis padres y hermana	Sin su esfuerzo y apoyo incondicional no hubiera logrado tan ardua tarea. Gracias por todo, por su amor, su apoyo, comprensión y motivación para seguir adelante. Hicieron un buen trabajo**.**
Mi esposa	Por su apoyo en la recta final de este trabajo y por hacerme la vida más feliz con su compañía.
Mi familia	Por sus muestras de apoyo durante todo este trayecto y por los momentos gratos que hemos vivido.
Mis amigos y sus familias	Por el apoyo que siempre nos dimos unos a otros, así como el amor con que cada familia nos recibió.
Mi bella patria Guatemala	Para que cada vez hagamos de ti una patria mejor.

Erick Orlando Villagrán Zelada

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES III
GLOSARIO V
RESUMEN IX
OBJETIVOS XI
INTRODUCCIÓN XIII

1. INTRODUCCIÓN AL SISTEMA 1
   1.1. Aplicación base 1
   1.2. Reportes 2

2. ESTRUCTURA DE DATOS 13
   2.1. Estructuras disponibles 14
   2.2. Selección de información 21

3. GUÍA PARA LA CARGA DE INFORMACIÓN 23
   3.1. Pasos para carga de información 23
      3.1.1. Paso 1. Generación de la información 23
      3.1.2. Paso 2. Carga de la información a las tablas temporales 25
      3.1.3. Paso 3. Revisión de la carga a las tablas temporales 36
      3.1.4. Paso 4. Carga de la información a las tablas de producción . 37
      3.1.5. Paso 5. Verificar que la carga fue exitosa 38
   3.2. *Sql Loader* vrs. tablas externas 39
   3.3. ETL puesto a prueba 40

4. TRANSFERENCIA DE CONOCIMIENTOS 43
4.1. Administración de *Oracle Discoverer* 43
4.1.1. Qué es *Oracle Discoverer* 43
4.1.2. Elementos fundamentales de *Oracle Discoverer* 43
4.2. Administración de grandes cantidades de información 46
4.2.1. Extracción de la información 46
4.2.2. Almacenamiento de la información 48
4.2.3. Análisis de la información 48
4.3. Particiones e índices 49
4.3.1. Qué es una partición 49
4.3.1.1. Particionamiento por rango 50
4.3.1.2. Particionamiento por lista de valores 50
4.3.1.3. Particionamiento por función *hash* (dispersión) . 50
4.3.1.4. Particionamiento por rango-lista 50
4.3.1.5. Particionamiento por rango-dispersión 51
4.3.2. Qué es un índice particionado 51
4.3.2.1. Índices particionados globales 52
4.3.2.2. Índices particionados locales 52

CONCLUSIONES 53
RECOMENDACIONES 55
BIBLIOGRAFÍA 57
APÉNDICE 59

ÍNDICE DE ILUSTRACIONES

FIGURAS

1. Cantidad de años para graduación 3
2. Reporte uso de salones 4
3. Asignación de cursos 5
4. Demografía de estudiantes 6
5. Historial de cursos por estudiante 7
6. Asignaciones por catedrático 8
7. Reprobados por curso 9
8. *Ranking* de estudiantes 10
9. Cantidad de graduados 11
10. Análisis de estudiantes por curso 12
11. Proceso de carga de la información 13

TABLAS

I. Estudiante 15
II. Asignacion sistemas 17
III. Área 18
IV. Catedrático 18
V. Subarea 19
VI. Salón 19
VII. Horario 20
VIII. Departamento 21
IX. Curso 21

X. Curso aprobado 22
XI. Aprobados sistemas 23
XII. Graduado 23
XIII. Características del servidor de prueba 41
XIV. Características de PC cliente 41

GLOSARIO

APPEND	Instrucción de *SQL*Loader* que indica que los registros se agregan a la tabla destino.
Data mart	Conjunto de datos que se relacionan entre si y que en conjunto forman un caso de estudio.
Data Warehouse	Conjunto de *datamarts* que pueden o no estar relacionados entre sí.
ETL	Extracción, trasformación y carga de información.
Índice	Estructura de almacenamiento que está asociada a una tabla, que permite acceder a un registro específico de forma inmediata.
Oracle	Base de datos número 1 del mundo, se caracteriza por manejar gran cantidad de información.
Oracle Discoverer	Herramienta que sirve para visualizar *datamarts*.

Partición	Estructura de tabla en la cual se puede dividir la información, un campo indica a qué partición pertenece.
Ps/sql	Lenguaje de programación de *Oracle*, el cual se utiliza para manipular la información.
RAID 5	Metodología de almacenamiento que indica que está todo protegido a fallas.
Sql	*Structured Query Lenguaje*, lenguaje de consultas, es para consultar datos directamente a la base de datos.
Sqlloader	Herramienta de *Oracle* que se utiliza para cargar la información a la base de datos, se utiliza cuando los datos están en archivos de texto.
SQLServer	Base de datos que pertenece a Microsoft, se caracteriza por ser de bajo costo.
Sybase IQ	Base de datos.
Terabyte	Unidad de medida que indica 1024 GB.

TRUNCATE	Instrucción de *SQL*Loader* que indica que primero se eliminan los registros de la tabla para luego agregar la data a la tabla destino.

RESUMEN

En la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala existe un *DataMart* con la información de los alumnos inscritos, el cual incluye: notas, asignación de cursos con sus respectivos salones, graduados, etc. Surge la necesidad de tener el *DataMart* actualizado, para resolver dicha situación.

El primer capítulo, describe en forma breve en qué consiste el sistema, dando énfasis en el diagrama del modelo estrella en que está basado el *DataMart,* así como mostrar las principales consultas que muestra el sistema.

En el capítulo dos, se describen las estructuras necesarias que debe generar la fuente de información, para que la carga sea posible.

En el capítulo tres, se describe cómo la carga de información se realiza manualmente, finalizando el capítulo con la comparación de dos técnicas de carga de *Oracle*, *sqlloader* y tablas de archivos.

Finalmente en el capítulo cuatro, se desarrolla una documentación de cómo administrar el producto final que se encuentra desarrollado en *Oracle Discoverer*. Debido a que la información crece en función del tiempo, dar un vistazo a la administración de grandes cantidades de data. Finalmente se muestra la mejor manera de acceder a la información, almacenando la data en tablas particionadas con índices.

En el apéndice, se describe el concepto de ETL (*Extraction, Transformation and Loading*), el cual fue el punto de partida para automatizar este proceso. Y también

describiremos las herramientas disponibles para la carga de información a una base de datos *Oracle*, así como el lenguaje de programación PL/SQL.

OBJETIVOS

General

Actualizar el *DataMart* hasta diciembre 2009, así como sentar las bases para la automatización de los procesos de carga de información para que el sistema contenga la información más actual posible, y así sea de gran utilidad para la toma de decisiones en beneficio de los estudiantes de la Facultad de Ingeniería.

Específicos

1. Contar con la información actualizada hasta diciembre 2009.

2. Definir los primeros pasos para automatizar los procesos de extracción y carga de la información.

3. Dejar una guía para la correcta actualización del *DataMart.*

INTRODUCCIÓN

El siglo XXI se caracteriza por la rápida evolución tecnológica lo que conlleva a muchas fuentes de información. Dónde tomar decisiones sin fundamentos puede llevar al fracaso. Es por ello que contar con sistemas para la toma de decisiones es clave para la buena gestión de las organizaciones. Estas herramientas proporcionan indicadores con los cuales se puede medir y con la ayuda de ellos poder determinar el rumbo de una organización en un momento oportuno.

En la Facultad de Ingeniera de la Universidad de San Carlos de Guatemala, ya existe un sistema para la toma de decisiones, desafortunadamente no cuenta con la documentación de los procesos de mantenimiento y actualización, con esta deficiencia impedía su constante actualización y como resultado, el sistema perdía su valor.

Es por ello que el presente trabajo surge de esta necesidad, orientándose a desarrollar la documentación de cómo actualizar dicho sistema. Para cumplir con este objetivo, este trabajo inicia con la recopilación de la información proveniente del Centro de Cálculo de la Facultad de Ingeniería, para luego cargarla y finalmente actualizar los análisis que previamente fueron creados.

1. INTRODUCCIÓN AL SISTEMA

1.1. Aplicación base

El presente trabajo es la continuación al realizado por otro grupo de estudiantes, quienes tuvieron como objetivo principal el desarrollo del *DataMart* académico, quedando pendiente una manera fácil de actualizar el *DataMart*. Dicho trabajo consistió en el diseño e implementación de la aplicación, utilizando las herramientas de inteligencia de negocios, *Oracle Discoverer*, sobre la plataforma *web Oracle Application Server*. Esta última herramienta se utiliza para montar las aplicaciones de *Oracle*, sobre un ambiente web, logrando con ésta, que cualquier persona con acceso a *internet* y un navegador, pueda fácilmente acceder a la información que tiene el *DataMart*.

El *DataMart* académico, básicamente tiene seis roles con los cuales es posible conectarse:

- Administrador
- Centro de cálculo
- Control
- Decanato
- Secretaría
- Sistemas

Cada uno de estos roles, tiene permiso según el área de interés de cada uno, ya que el *DataMart* se divide en los siguientes cubos:

- Asignaciones y notas
- Graduados
- Salones

1.2. Reportes

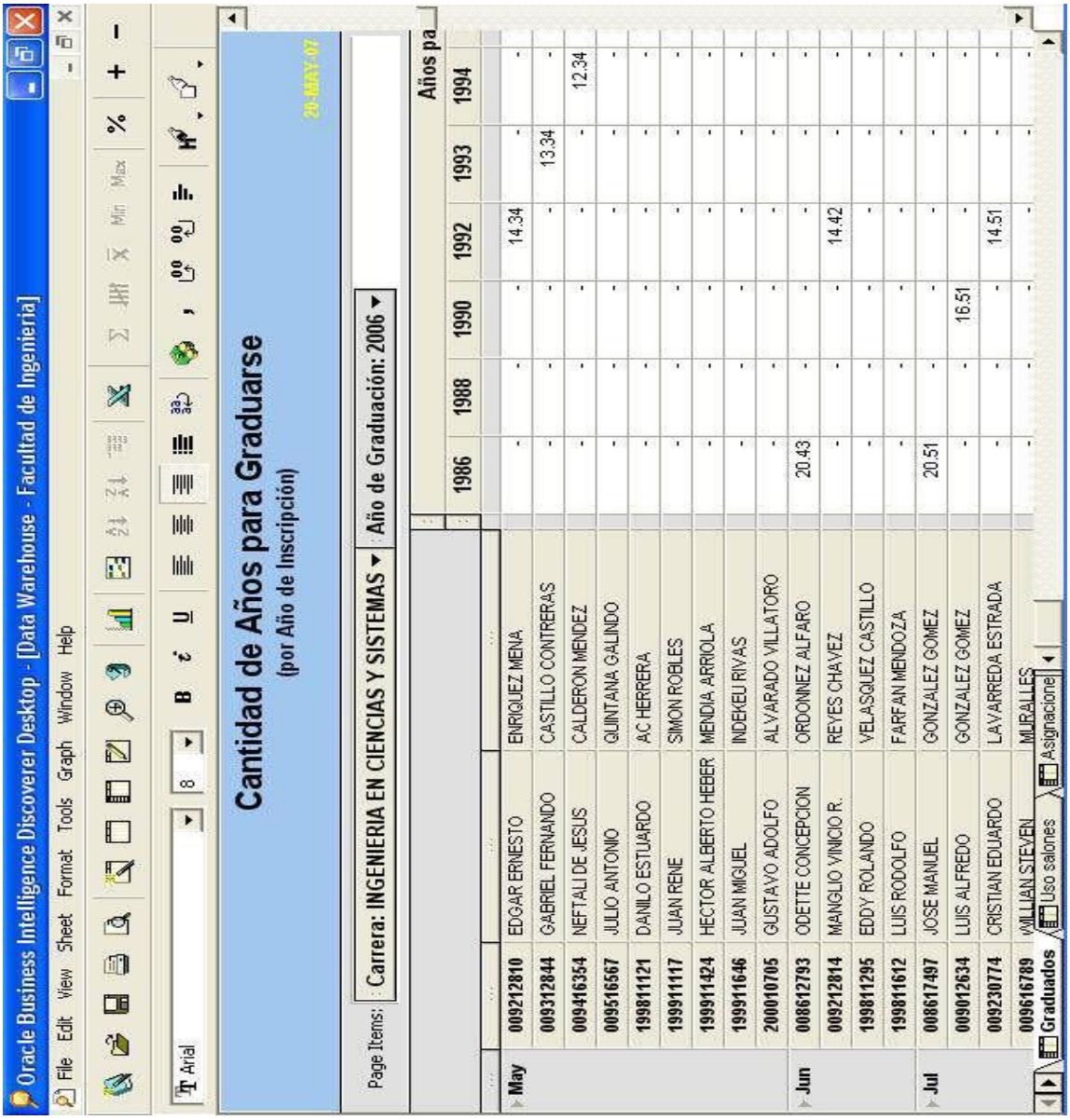
A continuación se muestran los principales reportes con los que actualmente cuenta la aplicación.

- Cantidad de años para graduarse
- Uso de salones
- Asignación de cursos
- Datos demográficos de estudiantes
- Historial de cursos por estudiante
- Asignaciones por catedrático
- Reprobados por curso
- *Ranking* de estudiantes
- Cantidad de graduados
- Estudiantes por curso y año de inscripción

Cantidad de años para graduación

Se muestra una estadística de la cantidad de años que le toma a un estudiante, graduarse de la carrera.

Figura 1. **Cantidad de años para graduación**

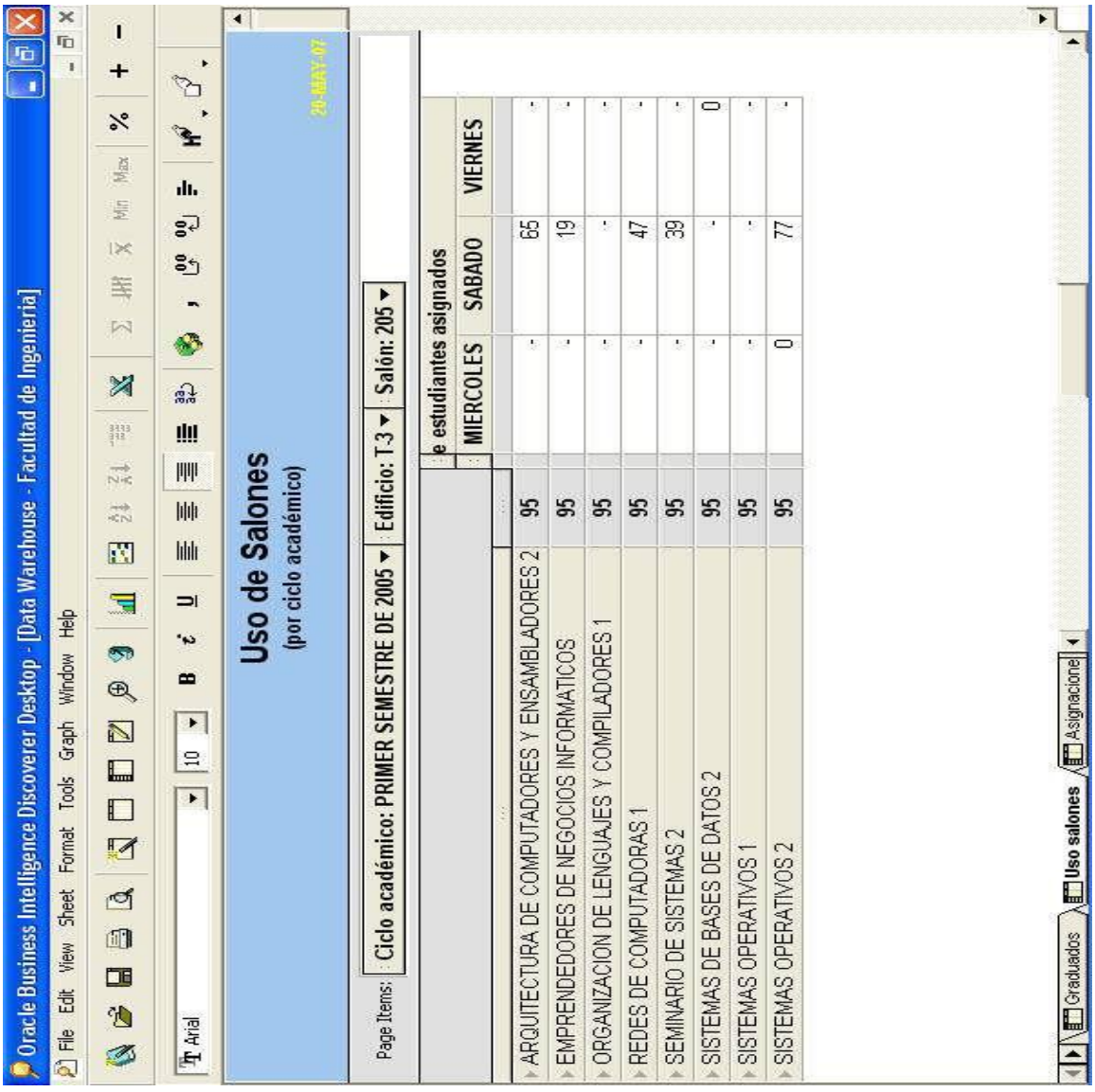


Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 67.

Reporte uso de salones

Este reporte muestra la capacidad y uso de los salones donde se imparten los cursos de sistemas. Con este reporte es posible saber que tan sub o sobre utilizado está siendo un salón.

Figura 2. **Reporte uso de salones**

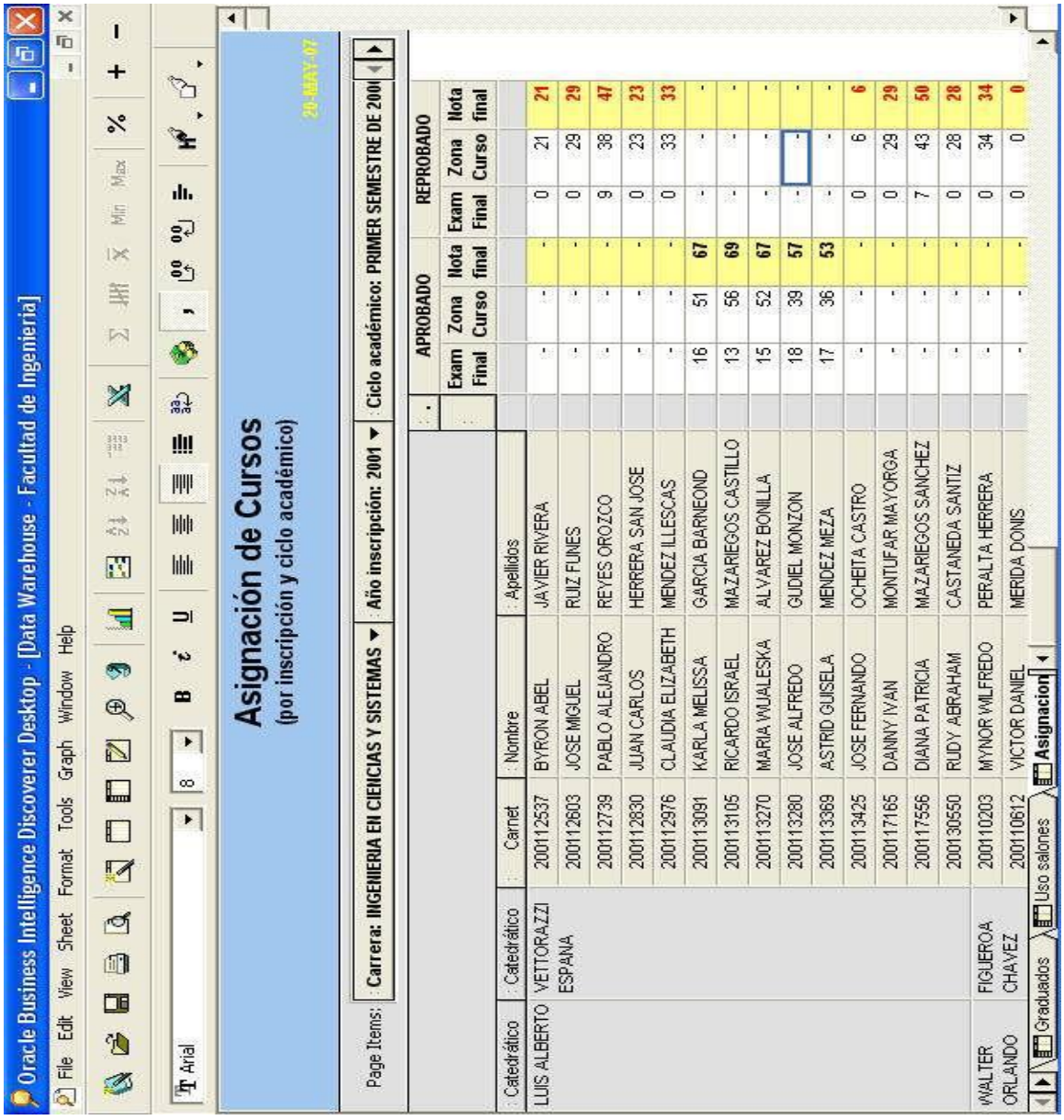


Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 68.

Asignación de cursos

Aquí es posible analizar la cantidad de estudiantes que se asignaron a determinado curso, así como también es posible analizar el porcentaje de aprobación en cada uno de ellos.

Figura 3. **Asignación de cursos**



Catedrático	Catedrático	Carnet	Nombre	Apellidos	APROBADO			REPROBADO		
					Exam Final	Zona Curso	Nota final	Exam Final	Zona Curso	Nota final
LUIS ALBERTO	VETTORAZZI ESPAÑA	200112537	BYRON ABEL	JAVIER RIVERA	-	-	-	0	21	21
		200112603	JOSE MIGUEL	RUIZ FUNES	-	-	-	0	29	29
		200112739	PABLO ALEJANDRO	REYES OROZCO	-	-	-	9	38	47
		200112830	JUAN CARLOS	HERRERA SAN JOSE	-	-	-	0	23	23
		200112976	CLAUDIA ELIZABETH	MENDEZ ILLESCAS	-	-	-	0	33	33
		200113091	KARLA MELISSA	GARCIA BARNEOND	16	51	67	-	-	-
		200113105	RICARDO ISRAEL	MAZARIEGOS CASTILLO	13	56	69	-	-	-
		200113270	MARIA WUALESKA	ALVAREZ BONILLA	15	52	67	-	-	-
		200113280	JOSE ALFREDO	GUDIEL MONZON	18	39	57	-	-	-
		200113369	ASTRID GUISELA	MENDEZ MEZA	17	36	53	-	-	-
		200113425	JOSE FERNANDO	OCHEITA CASTRO	-	-	-	0	6	6
		200117165	DANNY IVAN	MONTUFAR MAYORGA	-	-	-	0	29	29
		200117556	DIANA PATRICIA	MAZARIEGOS SANCHEZ	-	-	-	7	43	50
		200130550	RUDY ABRAHAM	CASTAÑEDA SANTIZ	-	-	-	0	28	28
WALTER ORLANDO	FIGUEROA CHAVEZ	200110203	MYNOR WILFREDO	PERALTA HERRERA	-	-	-	0	34	34
		200110612	VICTOR DANIEL	MERIDA DONIS	-	-	-	0	0	0

Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 69.

Demografía de estudiantes

Con este reporte es posible analizar los estudiantes según, su rango de edad y/o genero, según el año de inscripción .

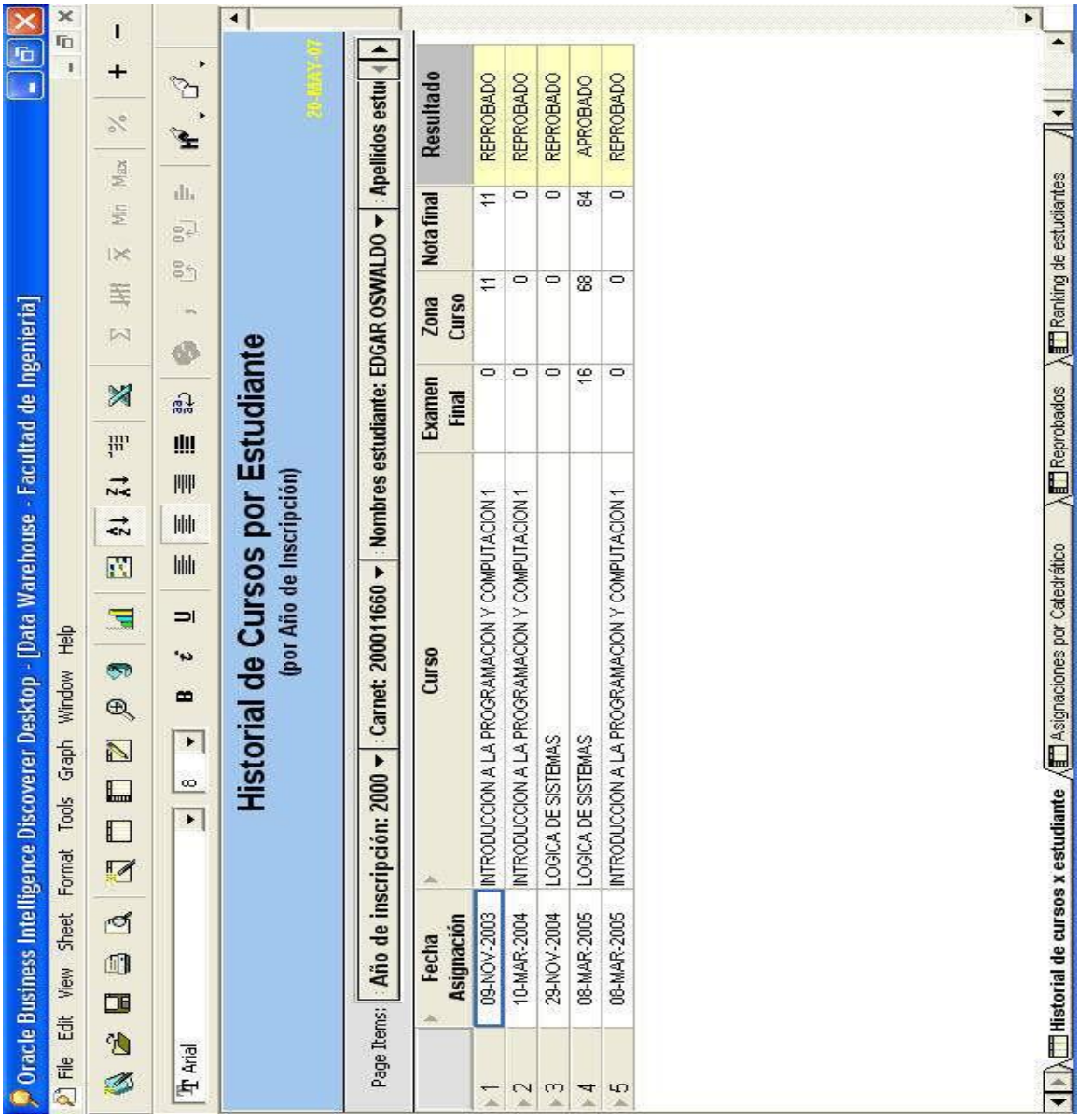
Figura 4. **Demografía de estudiantes**

Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 70.

Historial de cursos por estudiante

Este reporte presenta el historial de cursos asignados por un estudiante determinado. Se muestran todas las asignaciónes realizadas, así como también, si el curso fue aprobado o no, junto con la composición de la nota final obtenida.

Figura 5. **Historial de cursos por estudiante**

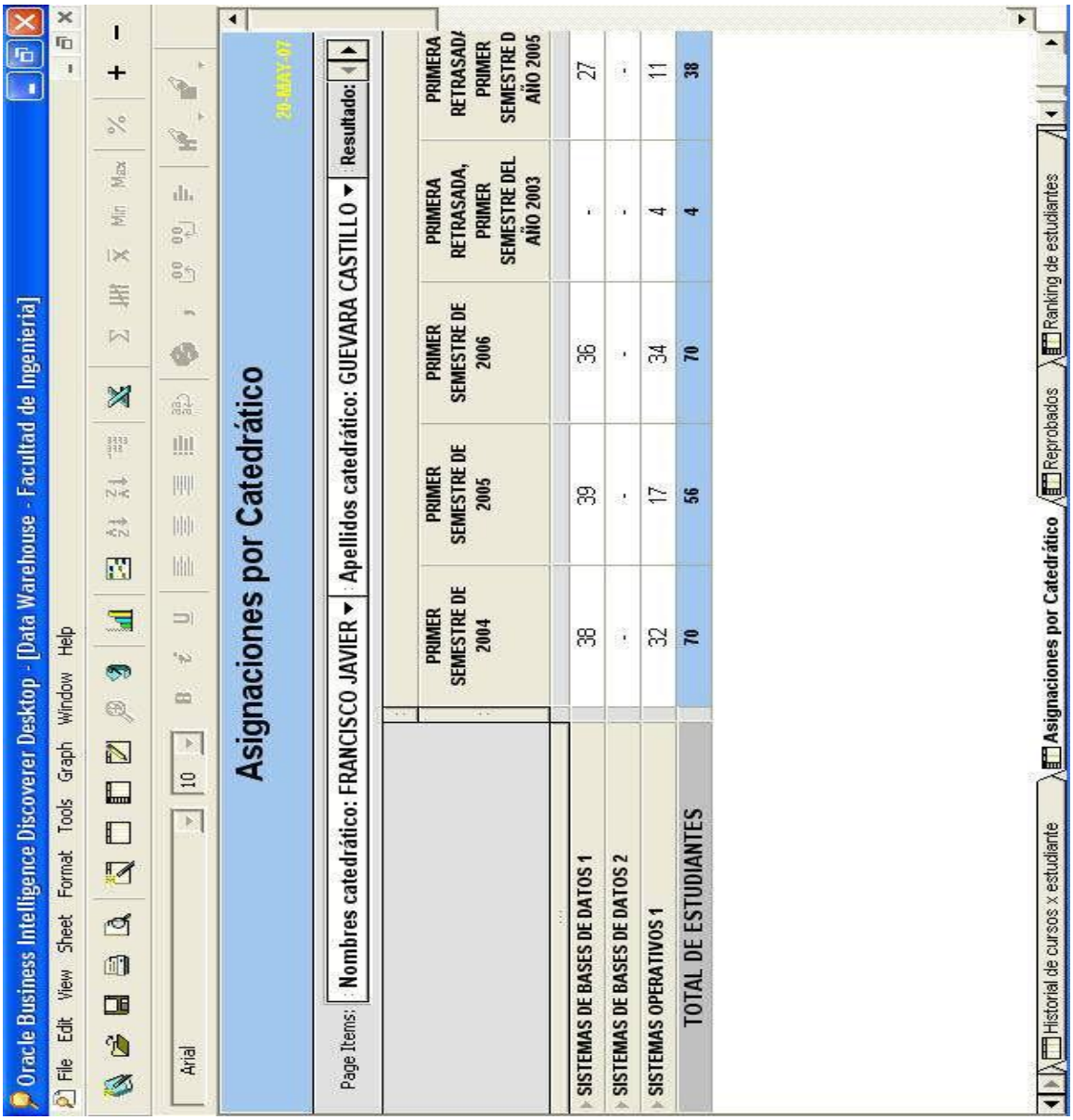


Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 71.

Asignaciones por catedrático

Este reporte presenta la cantidad de estudiantes asignados por catedrático y el curso impartido por el mismo, durante un ciclo. Aquí se puede analizar la cantidad de cursos que un catedrático está impartiendo durante cada uno de los ciclos académicos.

Figura 6. **Asignaciones por catedrático**

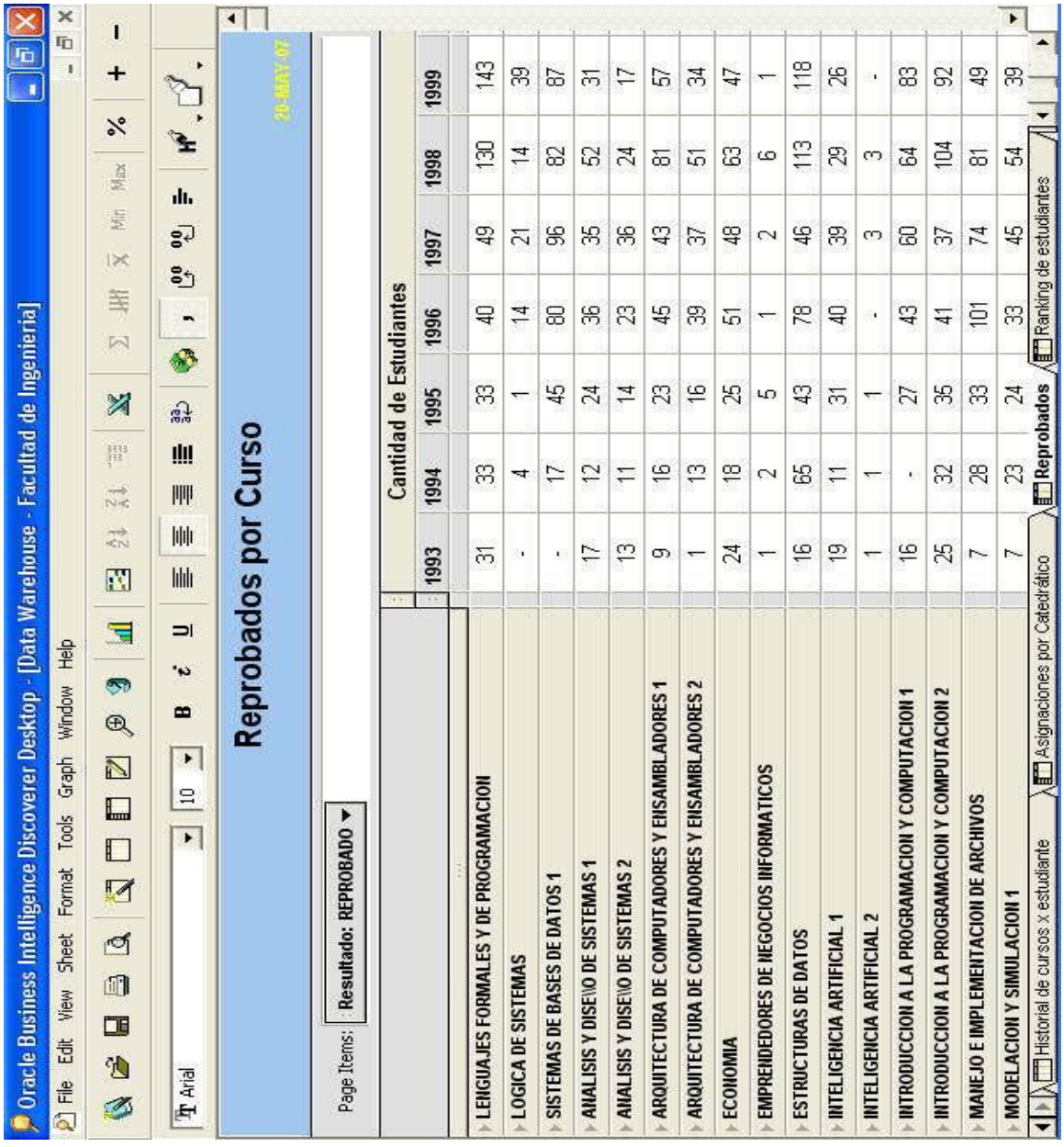


Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 72.

Reprobados por curso

Con este reporte es posible observar la cantidad de estudiantes reprobados durante un ciclo, para cada uno de los cursos que se imparten.

Figura 7. **Reprobados por curso**

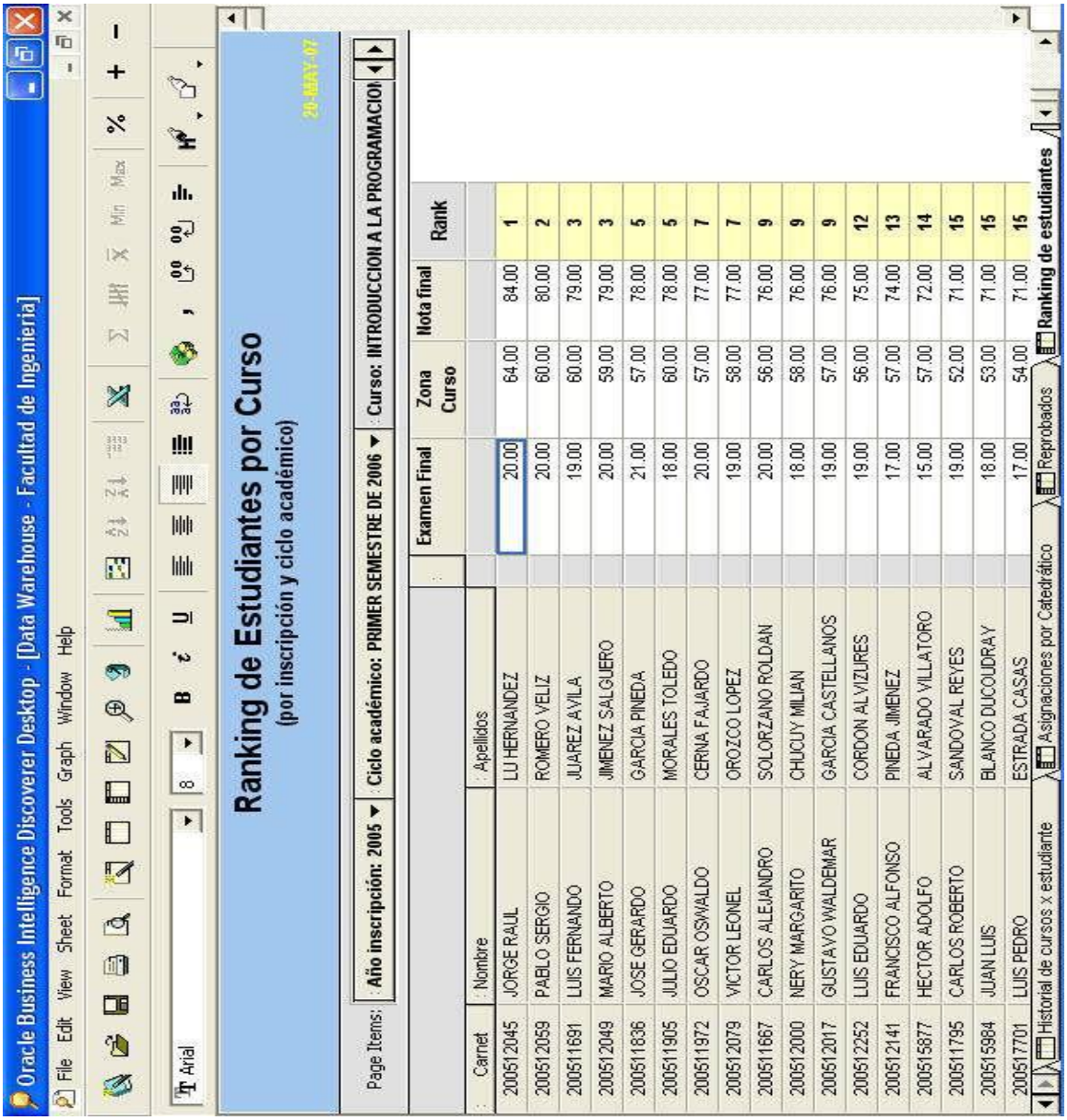


Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 73.

Rankin de estudiantes

Con este reporte es posible encontrar los estudiantes que tienen un mejor desempeño en sus notas. En contraparte, se pueden encontrar los que tienen muy bajo con respecto a sus calificaciones.

Figura 8. ***Rankin* de estudiantes**



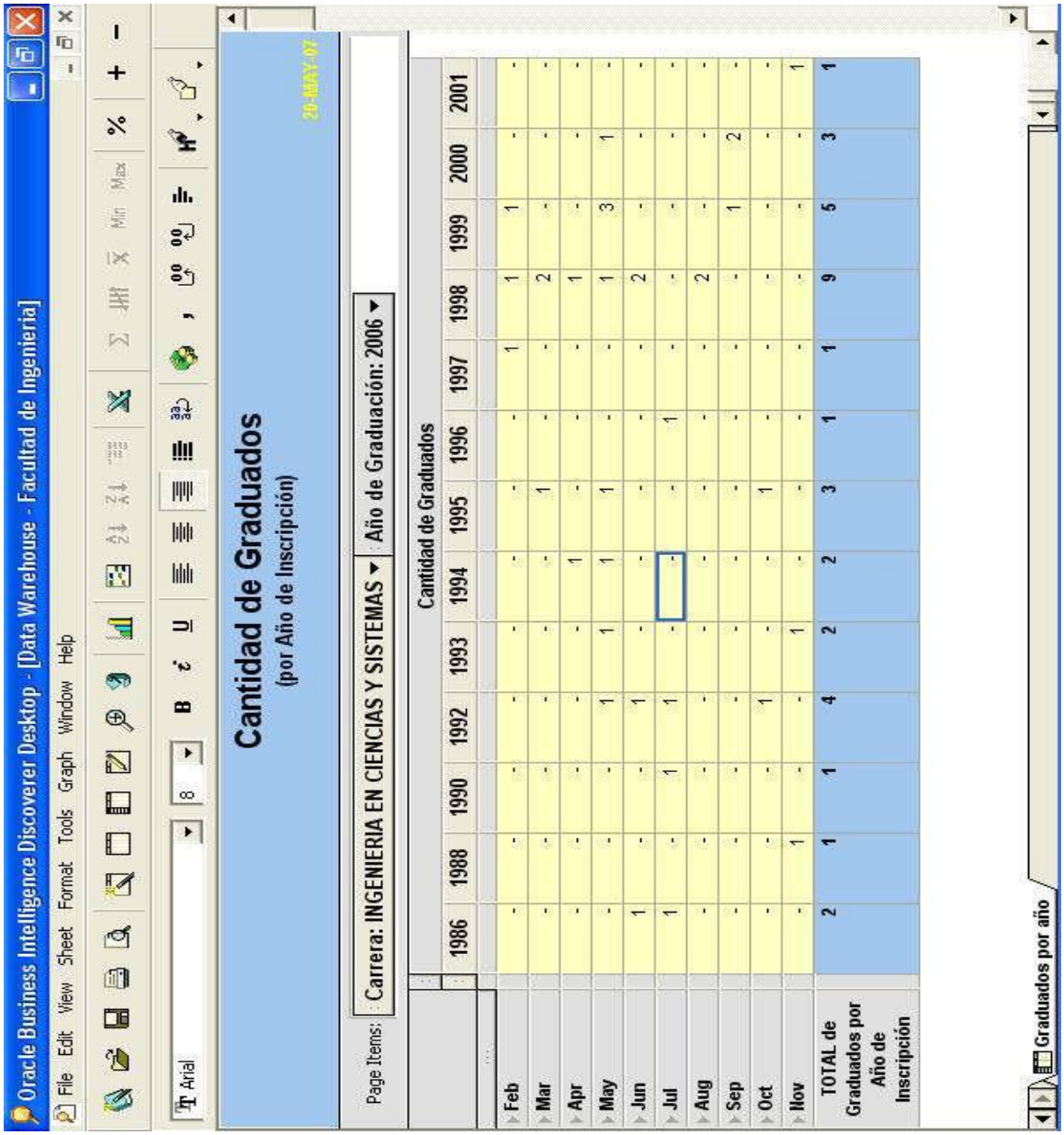
Carnet	Nombre	Apellidos	Examen Final	Zona Curso	Nota final	Rank
200512045	JORGE RAUL	LU HERNANDEZ	20.00	64.00	84.00	1
200512059	PABLO SERGIO	ROMERO VELIZ	20.00	60.00	80.00	2
200511691	LUIS FERNANDO	JUAREZ AVILA	19.00	60.00	79.00	3
200512049	MARIO ALBERTO	JIMENEZ SALGUERO	20.00	59.00	79.00	3
200511836	JOSE GERARDO	GARCIA PINEDA	21.00	57.00	78.00	5
200511905	JULIO EDUARDO	MORALES TOLEDO	18.00	60.00	78.00	5
200511972	OSCAR OSWALDO	CERNA FAJARDO	20.00	57.00	77.00	7
200512079	VICTOR LEONEL	OROZCO LOPEZ	19.00	58.00	77.00	7
200511667	CARLOS ALEJANDRO	SOLORZANO ROLDAN	20.00	56.00	76.00	9
200512000	NERY MARGARITO	CHUCUY MILIAN	18.00	58.00	76.00	9
200512017	GUSTAVO WALDEMAR	GARCIA CASTELLANOS	19.00	57.00	76.00	9
200512252	LUIS EDUARDO	CORDON ALVIZURES	19.00	56.00	75.00	12
200512141	FRANCISCO ALFONSO	PINEDA JIMENEZ	17.00	57.00	74.00	13
200515877	HECTOR ADOLFO	ALVARADO VILLATORO	15.00	57.00	72.00	14
200511795	CARLOS ROBERTO	SANDOVAL REYES	19.00	52.00	71.00	15
200515984	JUAN LUIS	BLANCO DUCOUDRAY	18.00	53.00	71.00	15
200517701	LUIS PEDRO	ESTRADA CASAS	17.00	54.00	71.00	15

Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 74.

Cantidad de graduados

Este reporte muestra la cantidad de estudiantes graducados, agrupados por la el año de inscripción.

Figura 9. **Cantidad de graduados**

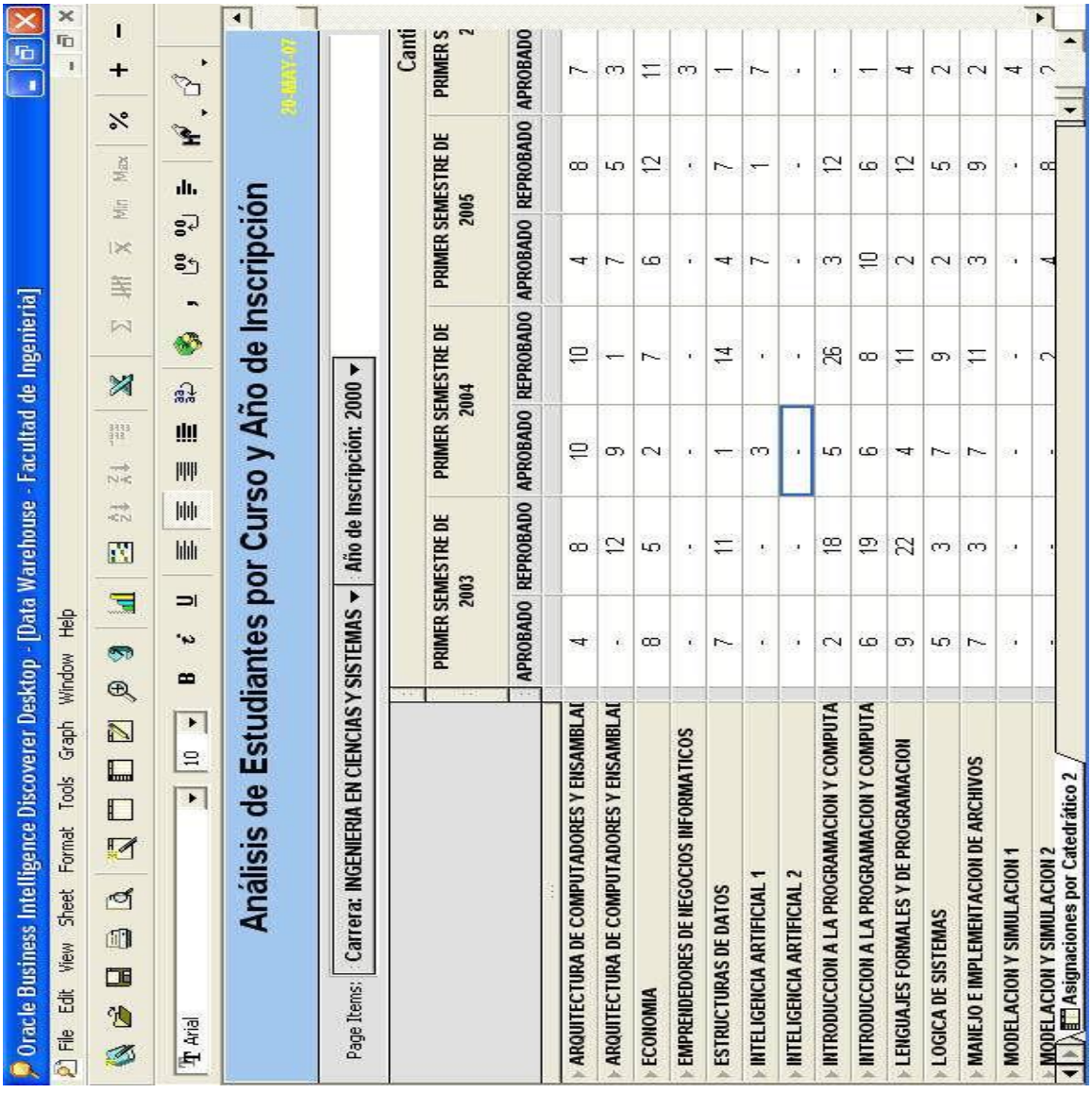


Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 75.

Analisis de estudiantes por curso

Con este reporte es posible analizar la cantidad de estudiantes aprobados y los reprobados de cada uno de los cursos de la carrera, durante cada uno de los ciclo educativos. Con este tipo de análisis es posible entender en qué cursos debe reforzarse la enceñanza, para reducir el nivel de estudiantes reprobados.

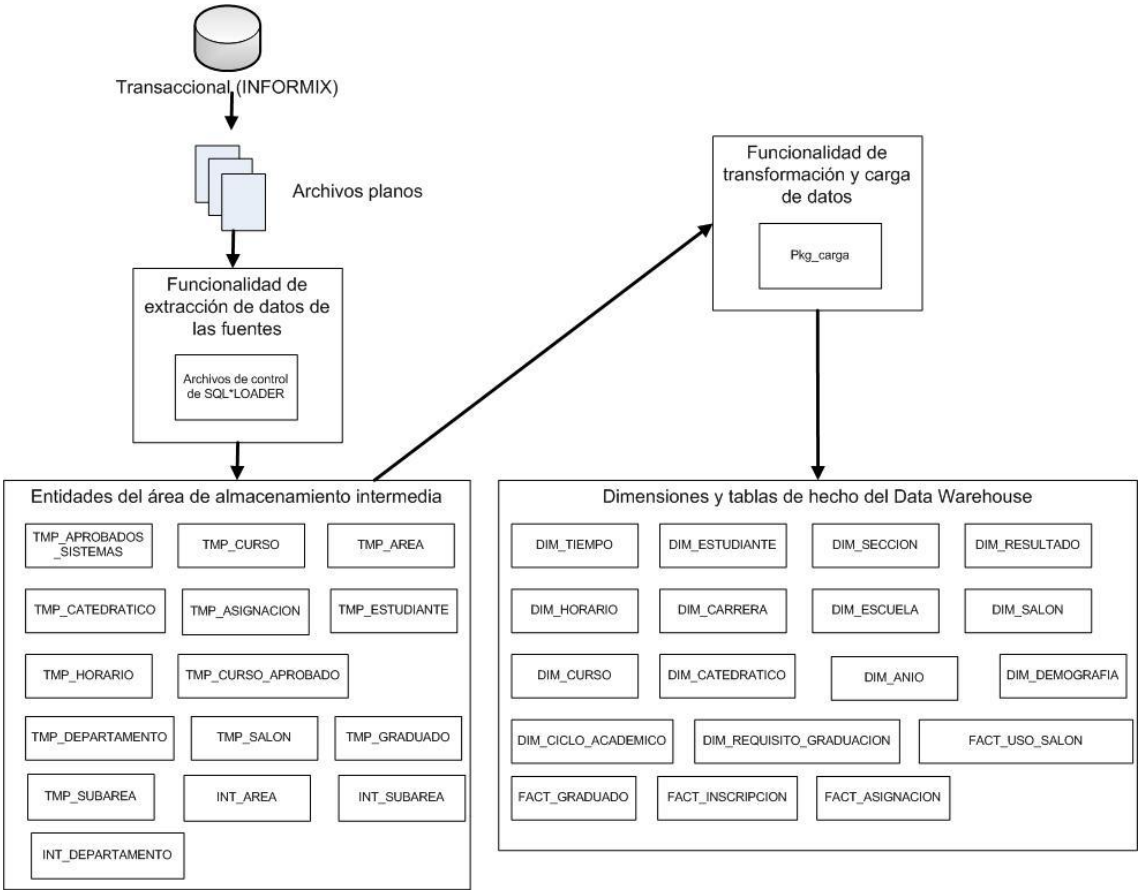
Figura 10. **Análisis de estudiantes por curso**



Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 76.

2. ESTRUCTURA DE DATOS

Figura 11. **Proceso de carga de la información**



Fuente: REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA, Pablo Augusto. Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la USAC. p. 52.

El proceso para actualizar un *DataMart* se puede dividir en dos partes:

- Definición de la estructura de los datos que son necesarios para la carga (en la figura de arriba es la parte de archivos planos que es la salida del sistema que está en INFORMIX que se verá en este capítulo).

- Cargar los datos al sistema para su procesamiento (en la figura son los procesos de extracción, transformación y carga que se verá en el siguiente capítulo).

2.1. Estructuras disponibles

La información se encuentra almacenada en una base de datos Informix, la cual tiene las tablas que se describen a continuación:

Estudiante

Contiene información general de estudiante como nombre, apellidos, dirección, localización geográfica teléfono, título, fecha de ingreso, cursos aprobados, promedio, etc.

Tabla I. **Estudiante**

ESTUDIANTE	Carnet
	Apellido
	Nombre
	Direst
	Colonia
	Coddept
	Codmun
	Zonapos
	Telest
	Codcarr
	Codins
	Sexo
	Codnac

Continuación tabla I.

	Estcivil
	Nacnac
	Depnac
	Munnac
	Fecnac
	Estproce
	Título
	Codest
	Fecing
	Concur
	Credapro
	Promgen
	Promant
	Currapro
	Curraproju
	Currequi
	Currsufi
	Currprob
	Anocon
	Promcurr
	Promcred
	Histins
	Email
	Suspendido
	Fecha

Fuente: elaboración propia.

Asignación Sistemas

Contiene la información de los cursos que un estudiante de sistemas se asigno en un periodo de tiempo (primer semestre, vacaciones primer semestre, 1era retrasada primer semestre, etc.).

Tabla II. **Asignación Sistemas**

ASIGNACION SISTEMAS	Carnet
	Codcurso
	Periodo
	Fecasig
	Sección
	Zona
	Exfinal
	Zonalab
	Feclab
	Codprob
	Codcarr
	Transacción

Fuente: elaboración propia.

Área

La Facultad tiene una división de los cursos en áreas, que sería el primer nivel de la división.

Tabla III. **Área**

ÁREA	Coddep
	Codarea
	Nomarea
	Regper
	Linkpre

Fuente: elaboración propia.

Catedrático

Contiene la información del personal docente que labora en la Facultad.

Tabla IV. **Catedrático**

CATEDRÁTICO	Regper
	Apellido
	Nombre
	Dirper
	Colonica
	Coddep
	Codmun
	Zonapos
	Telper
	Sexo
	Codnac
	Estcivil
	Nacnac
	Depnac
	Munnac
	FECOAC
	Título
	Codtitu
	Orden
	Registro

Fuente: elaboración propia.

Subárea

Segundo nivel de la división.

Tabla V. **Subárea**

SUBÁREA	Coddep
	Codarea
	Codsubarea
	Nomsubarea
	Regper

Fuente: elaboración propia.

Salón

Contiene la información de los diferentes salones que conforman la Facultad

Tabla VI. **Salón**

SALÓN	Codedifi
	Nomsalon
	Capsalon

Fuente: elaboración propia.

Horario

Contiene los diferentes horarios en que se imparten los cursos en un determinado período.

Tabla VII. **Horario**

HORARIO	Codedifi
	Nomsalon
	Tinicio
	Tfinal
	Regper
	Codcurso
	Sección
	Tiposec
	Totasig
	Lunes
	Martes
	Miércoles
	Jueves
	Viernes
	Sábado
	Período
	Límite
	Estado
	Año

Fuente: elaboración propia.

Departamento

Contiene la información del tercer nivel de clasificación.

Tabla VIII. **Departamento**

DEPARTAMENTO	Coddep
	Nomdep
	Regper
	Linkpre

Fuente: elaboración propia.

Curso

Contiene la información de los diferentes cursos que pertenecen al pensa de la carrera.

Tabla IX. **Curso**

CURSO	Codcurso
	Nomcurso
	Coddep
	Codarea
	Codsubarea
	Zona
	Alterna

Fuente: elaboración propia.

Curso aprobado

Contiene los cursos aprobados por todos los estudiantes.

Tabla X. **Curso Aprobado**

CURSO APROBADO	Carnet
	Codcarr
	Codcurso
	Fecapro
	Periapro
	Sección
	Formapro
	Zona
	Exfinal
	Posrel
	Intentos
	Créditos

Fuente: elaboración propia.

Aprobados Sistemas

Contiene los cursos aprobados por los estudiantes de la carrera de Ingeniería en Ciencias y Sistemas.

Tabla XI. **Aprobados Sistemas**

APROBADOS SISTEMAS	Carnet
	Codcarr
	Codcurso
	Fecapro

Continuación tabla XI.

	Periapro
	Sección
	Formapro
	Zona
	Zona
	Exfinal
	Posrel
	Intentos
	Créditos

Fuente: elaboración propia.

Graduado

Contiene los estudiantes que se han graduado de la carrera.

Tabla XII. **Graduado**

GRADUADO	Carnet
	Codcarr

Fuente: elaboración propia.

2.2. Selección de información

Debido a que el modelo de datos para carga del *DataMart* se basó en la estructura de las tablas anteriormente descritas, lo único que hace falta es generar la información de las tablas descritas, con la estructura ya definida. En este punto se tiene la peculiaridad de elegir varias maneras de generar la información: archivos planos, Excel, o en una base de datos de ACCESS, para realizar la carga de una manera rápida y

sin complicaciones se escogieron los archivos planos, con separador de campos al carácter “|” , debido a que en la información puede que se presenten comas (“,”), y esto haría difícil la carga de la información.

3. GUÍA PARA LA CARGA DE INFORMACIÓN

3.1. Pasos para carga de información

En todo sistema de información lo que se busca es que la información se encuentre disponible en cualquier momento y que ésta esté actualizada, además dicho proceso de actualización sea de forma automática y sin participación del ser humano, ahora bien cuando la información a recabar tiene muchos factores que impiden su recopilación a una fecha determinada, es necesario que una persona sea la encargada de generar la información, y como este es el caso para la carga de información del *DataMart* de la facultad de ingeniería es necesario definir el siguiente proceso:

3.1.1. Paso 1. Generación de la información

Para este paso es necesario que una persona genere varios archivos de texto con la siguiente estructura y nombres (los campos separados por el carácter "|"), esto se debe de hacer cuando toda la información del semestre que se desea cargar ya este completa y que no hayan cambios en el futuro.

Tabla: área
Campos: coddep, codarea, nomarea, regper, linkpre
Archivo: área.txt

Tabla: aprobados_sistemas
Campos: carnet, codcarr, codcurso, fecapro, periapro, seccion, formapro, zona, exfinal, posrel, intentos, créditos
Archivo: aprobados_sistemas.txt

Tabla: asignación_sistemas
Campos: carnet, codcurso, período, fecasig, sección, zona, exfinal, zonalab, feclab, codprob, codcarr, transacción
Archivo: asignación_sistemas.txt

Tabla: catedrático
Campos: regper, apellido, nombre, dirper, colonia, coddep, codmun, zonapos, telper, sexo, codnac, estcivil, nacnac, depnac, munnac, fecnac, titulo, codtitu, orden, registro
Archivo: catedratico.txt

Tabla: curso
Campos: codcurso, nomcurso, coddep, codarea, codsubarea, zona, alterna, descurso
Archivo: curso.txt

Tabla: curso_aprobado
Campos: carnet, codcarr, codcurso, fecapro, periapro, seccion, formapro, zona, exfinal, posrel, intentos, créditos

Tabla: departamento
Campos: coddep, nomdep, regper, linkpre
Archivo: departamento.txt

Tabla: estudiante
Campos: carnet, apellido, nombre, direst, colonia, coddept, codmun, zonapos, telest, codcarr, codins, sexo, codnac, estcivil, nacnac, depnac, munnac, fecnac, estproce, titulo, codest, fecing, concur, credapro, promgen, promant, currapro, curraproju, currequi, currsufi, currprob, anocon, promcurr, promcred, histins, email, suspendido, fecha
Archivo: estudiante.txt
Tabla: graduado

Campos: carnet, codcarr
Archivo: graduado.txt

Tabla: horario
Campos: codedifi, nomsalon, tinicio, tfinal, regper, codcurso, seccion, tiposec, totasig, lunes, martes, miércoles, jueves, viernes, sábado, período, límite, estado, anio
Archivo: graduado.txt

Tabla: salón
Campos: codedifi, nomsalon, capsalon
Archivo: salon.txt

Tabla: subárea
Campos: coddep, codarea, codsubarea, nomsubarea, regper
Archivo: subarea.txt

Estos archivos se deben poner en el servidor donde está el DataMart (192.168.18.18) en el directorio /carga_datos/

3.1.2. Paso 2. Carga de la información a las tablas temporales

Para este paso se utiliza un utilitario del *Oracle* que se llama *SQLoader*, este utiliza unos archivos de control que contienen la estructura de la tabla donde va a caer la data del archivo que se está trabajando, estos archivos están localizados en el directorio /carga_datos/ctls/.

area.ctl
OPTIONS (SKIP=1)
LOAD DATA
INTO TABLE "TMP_AREA"

```
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
  (CODDEP,
   CODAREA,
   NOMAREA,
   REGPER,
   LINKPRE)

aprobados_sistemas.ctl
OPTIONS (SKIP=1)
LOAD DATA
INTO TABLE "TMP_APROBADOS_SISTEMAS"
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
  (CARNET,
   CODCARR,
   CODCURSO,
   FECAPRO,
   PERIAPRO,
   SECCIÓN,
   FORMAPRO,
   ZONA,
   EXFINAL,
   POSREL,
   INTENTOS,
   CRÉDITOS)
```

asignacion_sistemas.ctl

```
OPTIONS (SKIP=1)
LOAD DATA
INTO TABLE "TMP_ASIGNACION_SISTEMAS"
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
  (CARNET,
   CODCURSO,
   PERÍODO,
   FECASIG,
   SECCIÓN,
   ZONA,
   EXFINAL,
   ZONALAB,
   FECLAB,
   CODPROB,
   CODCARR,
   TRANSACCIÓN)
```

catedratico.ctl

```
OPTIONS (SKIP=1)
LOAD DATA
INTO TABLE "TMP_CATEDRATICO"
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
  (REGPER,
   APELLIDO,
```

```
  NOMBRE,
  DIRPER,
  COLONICA,
  CODDEP,
  CODMUN,
  ZONAPOS,
  TELPER,
  SEXO,
  CODNAC,
  ESTCIVIL,
  NACNAC,
  DEPNAC,
  MUNNAC,
  FECNAC,
  TÍTULO,
  CODTITU,
  ORDEN,
  REGISTRO)
```

curso.ctl

```
OPTIONS (SKIP=1)
LOAD DATA
INTO TABLE "TMP_CURSO"
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
  (CODCURSO,
  NOMCURSO,
  CODDEP,
```

```
  CODAREA,
  CODSUBAREA,
  ZONA,
  ALTERNA)
```

curso_aprobado.ctl

```
LOAD DATA
INTO TABLE "TMP_CURSO_APROBADO"
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY "" AND ""
TRAILING NULLCOLS
 (CARNET,
  CODCARR,
  CODCURSO,
  FECAPRO,
  PERIAPRO,
  SECCION,
  FORMAPRO,
  ZONA,
  EXFINAL,
  POSREL,
  INTENTOS,
  CREDITOS)
```

departamento.ctl

```
OPTIONS (SKIP=1)
LOAD DATA
INTO TABLE "TMP_DEPARTAMENTO"
FIELDS TERMINATED BY '|'
```

```
 OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
  (CODDEP,
   NOMDEP,
   REGPER,
   LINKPRE)

graduado.ctl
OPTIONS (SKIP=1)
LOAD DATA
INTO TABLE "TMP_GRADUADO"
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
  (CARNET,
   CODCARR)

estudiante.ctl
OPTIONS (SKIP=1)
LOAD DATA
INTO TABLE "TMP_ESTUDIANTE"
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
  (CARNET,
   APELLIDO,
   NOMBRE,
   DIREST,
   COLONIA,
```

CODDEPT,
CODMUN,
ZONAPOS,
TELEST,
CODCARR,
CODINS,
SEXO,
CODNAC,
ESTCIVIL,
NACNAC,
DEPNAC,
MUNNAC,
FECNAC,
ESTPROCE,
TÍTULO,
CODEST,
FECING,
CONCUR,
CREDAPRO,
PROMGEN,
PROMANT,
CURRAPRO,
CURRAPROJU,
CURREQUI,
CURRSUFI,
CURRPROB,
ANOCON,
PROMCURR,
PROMCRED,

```
  HISTINS,
  EMAIL,
  SUSPENDIDO,
  FECHA)

horario.ctl
OPTIONS (SKIP=1) LOAD DATA
INTO TABLE "TMP_HORARIO"
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY "" AND ""
TRAILING NULLCOLS
  (CODEDIFI,
  NOMSALON,
  TINICIO,
  TFINAL,
  REGPER,
  CODCURSO,
  SECCIÓN,
  TIPOSEC,
  TOTASIG,
  LUNES,
  MARTES,
  MIÉRCOLES,
  JUEVES,
  VIERNES,
  SÁBADO,
  PERÍODO,
  LIMITE,
  ESTADO,
```

```
  ANIO)
```

salon.ctl

```
OPTIONS (SKIP=1)
LOAD DATA
INTO TABLE "TMP_SALON"
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
  (CODEDIFI,
   NOMSALON,
   CAPSALON)
```

subarea.ctl

```
OPTIONS (SKIP=1)
LOAD DATA
INTO TABLE "TMP_SUBAREA"
APPEND
FIELDS TERMINATED BY '|'
 OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
  (CODDEP,
   CODAREA,
   CODSUBAREA,
   NOMSUBAREA,
   REGPER)
```

Se debe tomar encuentra que las tablas deben estar vacías, de lo contrario da error, si hay data y se quiere conservar hay que agregar la palabra APPEND, si se quiere

que en cada carga se borre la información que contienen hay que agregar la palabra TRUNCATE.

Cuando ya se tiene la información se procede a ejecutar el siguiente archivo sh.

carga.sh

```
sqlldr dwacademico/usac control=/carga_datos/ctls/curso.ctl data=/carga_datos/curso.txt
log=/carga_datos/ctls/curso.log bad=/carga_datos/ctls/curso.bad
discard=/carga_datos/ctls/curso.dsd

sqlldr dwacademico/usac control=/carga_datos/ctls/asignacion_sistemas.ctl
data=/carga_datos/asignacion_sistemas.txt
log=/carga_datos/ctls/asignacion_sistemas.log
bad=/carga_datos/ctls/asignacion_sistemas.bad
discard=/carga_datos/ctls/asignacion_sistemas.dsd

sqlldr dwacademico/usac control=/carga_datos/ctls/departamento.ctl
data=/carga_datos/departamento.txt log=/carga_datos/ctls/departamento.log
bad=/carga_datos/ctls/departamento.bad discard=/carga_datos/ctls/departamento.dsd

sqlldr dwacademico/usac control=/carga_datos/ctls/graduado.ctl
data=/carga_datos/graduado.txt log=/carga_datos/ctls/graduado.log
bad=/carga_datos/ctls/graduado.bad discard=/carga_datos/ctls/graduado.dsd

sqlldr dwacademico/usac control=/carga_datos/ctls/salon.ctl data=/carga_datos/salon.txt
log=/carga_datos/ctls/salon.log bad=/carga_datos/ctls/salon.bad
discard=/carga_datos/ctls/salon.dsd
```

```
sqlldr dwacademico/usac control=/carga_datos/ctls/area.ctl data=/carga_datos/area.txt
log=/carga_datos/ctls/area.log bad=/carga_datos/ctls/area.bad
discard=/carga_datos/ctls/area.dsd

sqlldr dwacademico/usac control=/carga_datos/ctls/catedratico.ctl
data=/carga_datos/catedratico.txt log=/carga_datos/ctls/catedratico.log
bad=/carga_datos/ctls/catedratico.bad discard=/carga_datos/ctls/catedratico.dsd

sqlldr dwacademico/usac control=/carga_datos/ctls/curso_aprobado.ctl
data=/carga_datos/curso_aprobado.txt log=/carga_datos/ctls/curso_aprobado.log
bad=/carga_datos/ctls/curso_aprobado.bad
discard=/carga_datos/ctls/curso_aprobado.dsd

sqlldr dwacademico/usac control=/carga_datos/ctls/estudiante.ctl
data=/carga_datos/estudiante.txt log=/carga_datos/ctls/estudiante.log
bad=/carga_datos/ctls/estudiante.bad discard=/carga_datos/ctls/estudiante.dsd

sqlldr dwacademico/usac control=/carga_datos/ctls/horario.ctl
data=/carga_datos/horario.txt log=/carga_datos/ctls/horario.log
bad=/carga_datos/ctls/horario.bad discard=/carga_datos/ctls/horario.dsd

sqlldr dwacademico/usac control=/carga_datos/ctls/subarea.ctl
data=/carga_datos/subarea.txt log=/carga_datos/ctls/subarea.log
bad=/carga_datos/ctls/subarea.bad discard=/carga_datos/ctls/subarea.dsd
```

3.1.3. Paso 3. Revisión de la carga a las tablas temporales

Luego se revisan los archivos .log para ver que todos los registros se insertaron

SQL*Loader: Release 10.2.0.1.0 - Production on Fri May 2 14:31:45 2008

Control File: c:\pasa\usac\carga\ctls\area.ctl
Data File: c:\pasa\usac\carga\area.txt
Bad File: c:\pasa\usac\carga\ctls\area.bad
Discard File: c:\pasa\usac\carga\ctls\area.dsd
(Allow all discards)

Number to load: ALL
Number to skip: 1
Errors allowed: 50
Bind array: 64 rows, maximum of 256000 bytes
Continuation: none specified
Path used: Conventional
Table "TMP_AREA", loaded from every logical record.
Insert option in effect for this table: INSERT
TRAILING NULLCOLS option in effect

Column Name	Position	Len	Term	Encl	Datatype
CODDEP	FIRST	*	|	O(") O(")	CHARACTER
CODAREA	NEXT	*	|	O(") O(")	CHARACTER
NOMAREA	NEXT	*	|	O(") O(")	CHARACTER

```
REGPER          NEXT     *     |       O(")  CHARACTER
                                       O(")
LINKPRE         NEXT     *     |       O(")  CHARACTER
                                       O(")
```

Table "TMP_AREA":

68 Rows successfully loaded.

0 Rows not loaded due to data errors.

0 Rows not loaded because all WHEN clauses were failed.

0 Rows not loaded because all fields were null.

Space allocated for bind array: 82560 bytes(64 rows)

Read buffer bytes: 1048576

Total logical records skipped: 1

Total logical records read: 68

Total logical records rejected: 0

Total logical records discarded: 0

Run began on Fri May 02 14:31:45 2008

Run ended on Fri May 02 14:31:46 2008

Elapsed time was: 00:00:00.19

CPU time was: 00:00:00.04

3.1.4. Paso 4. Carga de la información a las tablas de producción

Al cerciorarse que todo haya cargado, se procede de la siguiente manera: Abrir una sesión de SQL*PLUS con el usuario DWACADEMICO.

Borrar las tablas de bitácora con la siguiente instrucción:

```
Delete ctl_bitacora;
Delete ctl_error_carga
```

Ejecutar el proceso con los parámetros deseados:

SQL > EXECUTE PKG_CARGA.CARGAR_DATA_WAREHOUSE
(p_anio, p_semestre);

En donde los parámetros p_anio y p_semestre pueden tomar los siguientes valores:

- p_semestre: 1 = primer semestre, 2 = segundo semestre.
- a_anio: es el año completo como 2005, 2006 ó 2007.

Al terminar de correr el procedimiento se revisan las tablas ctl_bitacora y ctl_error_carga, si hubiera algún error poder eliminar todo lo cargado por el proceso con la siguiente instrucción:

Pkg_carga.limpiar_facts_periodo(p_anio , p_semestre);

3.1.5. Paso 5. Verificar que la carga fue exitosa

En este punto es donde entra el ser humano a verificar que todo el proceso anterior no haya tenido algún inconveniente se siguen los siguientes pasos:

a. Verificar las tablas descritas anteriormente (ctl_bitacora y ctl_error_carga)
b. Ejecutar el siguiente query:

```
select b.id, B.NOMBRE, count(1)
from usac.fact_asignacion a, usac.dim_ciclo_academico b
where  A.ID_CICLO_ACADEMICO = B.ID
and B.ANIO = '2006'—año de carga
group by b.id,  b.nombre
order by 1
```

Si están todos los períodos que se debieron cargar pasar al siguiente punto

c. Tener a la mano un certificado de cursos aprobados de un estudiante del año, y semestre de carga, para verificar que sí está todo cargado.

d. Revisar las consultas que muestren información del año y semestre de carga.

3.2. *Sql Loader* vrs. Tablas externas

Oracle provee otra forma de carga de información cuando dicha información está en archivos planos, esta se llama tablas externas.

En la práctica una tabla externa no es más que un archivo plano con cierto *layout* predefinido en el cual los campos están separados por algún carácter como una coma, comillas, o tabuladores, muy parecido al que se usaría para hacer una carga desde *SQL*Loader*, que es declarado en *Oracle* y que puede ser manejado directamente como cualquier otro objeto de la base de datos, pero a fin de cuentas es un archivo en el sistema operativo, con algunas limitantes, como no poder realizar *inserts, deletes, updates* sobre el objeto, no se pueden generar índices y su tamaño máximo es de 2GB.

Si el archivo a utilizar contiene muchos, es conveniente mejor usar *SQL*Loader* para previamente cargar los datos a *Oracle*, ya que los tiempos para consultas a estas tablas con bastante más grandes que los tiempos de una tabla normal, esto debido a que no se pueden crear índices sobre la tabla externa o, en su defecto, usar una segunda tabla, que sea llenada con la información de la externa con un simple *insert as select*, la cual tenga sus debidos índices.

Usos prácticos

Las tablas externas tienen muchos usos prácticos, los cuales se pueden colocar en dos categorías: proceso de negocio y administración de la base de datos.

Desde el punto de vista de negocio-proceso, las tablas externas responden a una necesidad vital en un ambiente de dato-almacenamiento, en el cual la extracción, transformación y los procesos de la carga son comunes. Las tablas externas hacen innecesario que los usuarios creen las tablas temporales durante estos procesos, reduciendo el espacio requerido y el riesgo de trabajos fallados.

Las tablas externas se pueden utilizar en vez de las tablas temporales y de utilidades como *SQL*Loader*. También proporcionan una manera fácil para que las compañías puedan cargar diversas fuentes de información en *Oracle* desde Excel, ACT!, o Access.

En cuanto a la administración de la base de datos. Si se desea supervisar los archivos que se revisan con más frecuencia como *alert.log* e *init.ora* desde un *prompt* SQL>. Entonces se pueden utilizar comandos del SQL para preguntar datos del archivo y de especificar criterios para un proceso más sofisticado.

Administración de la Base de Datos

Es importante saber qué vistas en *Oracle* contienen la información que pertenece a las tablas externas. La vista DBA_TABLES demuestra las tablas externas y tiene un valor de 0 para PCT_FREE, PCT_USED, INI_TRANS, y MAX_TRANS. El resto de las columnas del almacenaje en la vista son nulas. Los scripts que utiliza esta vista para determinar problemas deben ser actualizados para tener acceso a DBA_EXTERNAL_TABLES. Esta vista contiene todos los parámetros que se especificó cuando se creó la tabla externa.

3.3. ETL puesto a prueba

ETL son las siglas en inglés de Extraer, Transformar y Cargar (*Extract, Transform and Load*). Es el proceso que permite a las organizaciones mover datos desde múltiples fuentes, reformatearlos y limpiarlos, y cargarlos en otra base de datos,

DataMart o Data Warehouse para analizar, o en otro sistema operacional para apoyar un proceso de negocio.

Para demostrar el uso de un ETL, se muestra como puede reducir el tiempo de ejecución de un proceso, el cual tarda aproximadamente 10 horas. El proceso consiste en aproximadamente 30 millones de registros, a cada uno agregarle dos campos descriptivos, después generar un resumen agrupado, el cual da como resultado aproximadamente 50 mil registros, la maquina donde se ejecuta tiene las siguientes características:

TablaXIII. **Características del servidor de prueba**

Maquina	IBM P5
Procesador	4 DUALCORE
SO	Linux Red Hat
BD	*Oracle* 10g
Memoria	2 GB
Almacenamiento	3 TB

Fuente: elaboración propia.

El servidor donde se ejecuta el nuevo proceso tiene la siguiente configuración:

Tabla XIV. **Características de PC cliente**

Maquina	HP
Procesador	2 DUALCORE
SO	Windows
BD	
Memoria	2 GB
Almacenamiento	500 GB

Fuente: elaboración propia.

El tiempo de ejecución del proceso con el ETL tardo 1 hora, cabe mencionar que en el momento de la ejecución el procesador de la máquina llegó al 100%, y la memoria también. Esto demuestra que se necesita una fuerte inversión en hardware para implantar este tipo de soluciones, pero es un gran beneficio en tiempo de ejecución.

4. TRANSFERENCIA DE CONOCIMIENTOS

4.1. Administración de *Oracle Discoverer*

4.1.1. Qué es *Oracle Discoverer*

Oracle Discoverer es una potente herramienta intuitiva de reportes, análisis, y publicación Web que brinda fácil acceso a la información contenida en bases de datos, sin requerir amplios conocimientos de sistemas. Esto es posible tomar rápidas, mejores e informadas decisiones a todo nivel en las organizaciones. La herramienta muestra al usuario final, una vista gerencial de las complejas estructuras de bases de datos, permitiéndole enfocarse en el análisis de la información y no en cómo está la información.

4.1.2. Elementos fundamentales de *Oracle Discoverer*

Antes de diseñar o implementar un sistema *Oracle Discoverer*, necesitamos familiarizarnos con algunos conceptos:

- EUL

Es una colección de aproximadamente 50 tablas en la base de datos, las cuales son las únicas tablas que *Oracle Discoverer* modifica, ya que el resto de acceso a la base de datos, es únicamente de lectura.

El EUL, permite ver de una manera fácil y sencilla, las complejas estructuras de base de datos, lo que brinda un fácil acceso a la información.

- Business áreas (áreas de negocio)

Típicamente los usuarios finales están interesados un una porción de información de la base de datos, especialmente la que está relacionada con su área de trabajo. Utilizando el Administrador de *Oracle Discoverer*, es posible hacer una o más "*Business Áreas*" como contenedores de información relacionada.

Habiendo creado el *Business Area*, es posible cargar una o más tablas de base de datos que están relacionadas con la esa *Business Area*.

- Folders e Ítems

Las tablas que son cargadas en los *Business Áreas*, son representadas como folder párale usuario final. Las columnas de esas tablas, se muestran como ítems.

A menudo los nombres de las tablas y columnas de las bases de datos, son incomprensibles para el usuario final. Utilizando el Administrador de *Oracle Discoverer* es posible cambiar a un nombre más entendible para todos. Adicionalmente es posible crear nuevos ítems por medio de fórmulas y/o transformaciones a partir de ítems ya existentes.

- *Workbooks* y *Worksheets*

Los usuarios finales de *Oracle Discoverer*, analizan información utilizando *Folder* e Ítems para buscar información de su interés. Los *Worksheets* están agrupados dentro de los Workbooks y estos pueden ser almacenados directamente en la base de datos o en su computadora personal.

Con el Administrador de *Oracle Discoverer* es posible decidir qué *Workbook* puede abrir qué usuario o grupo de usuarios.

- Jerarquías y *Drills*

Jerarquías son relaciones lógicas entre ítems que permiten al usuario ver información sumarizada en más y en menos detalle. Para analizar información efectivamente, *Oracle Discoverer* permite al usuario final lo siguiente:

  - *Drill Down* permite ver más detalle de un ítem en particular (por ejemplo el total de ventas de una determinada región).
  - *Drill Up* muestra como el detalle de información contribuye para mostrar información un más alto nivel.

Cuando las tablas de base de datos son cargadas en los *Business Áreas, Oracle Discoverer* crea automáticamente las jerarquías para los tipos de dato fecha, lo que permite visualizar la información en días, semanas, meses, trimestres, años, etc.

- Folder Sumarios

Folder sumarios, son representaciones de información previamente cargadas y grabadas para su posterior reutilización. Estos sumarios se crean normalmente para mejorar el tiempo de respuesta para los usuarios finales.

4.2. Administración de grandes cantidades de información

En la actualidad el funcionamiento de una organización (empresas, compañías, bancos, universidades, gobiernos, etc.) tiene como resultado gran cantidad de información, la cual es provista por una gran cantidad de sistemas heterogéneos o no. El reto es lograr extraer esta gran cantidad de información, que las diferentes partes encajen como si fueran un engranaje, almacenarla, analizarla para encontrar tendencias y tomar decisiones administrativas *Data Warehouse*.

4.2.1. Extracción de la información

Toda organización tiene diferentes sistemas de información, tenemos por ejemplo; el de facturación, el de contabilidad, el de manufacturación, en las universidades el control académico, planilla etc., estos sistemas están construidos sobre diferentes bases de datos en el mejor de los casos, en otros son archivos, o simplemente no existen en medio magnético solo el papel, si toda la organización tiene solo un sistema para funcionar, perfecto, pero en la mayoría de los casos no es así, se tiene que encontrar la forma de cada sistema nos provea su información de una manera que sea fácil y sin ningún contratiempo reunirla en un solo lugar. En casos extremos se tiene la necesidad de emplear técnicas para extraer la información de documentos, escaneo y parseo, por medio de patrones identificar los datos deseados.

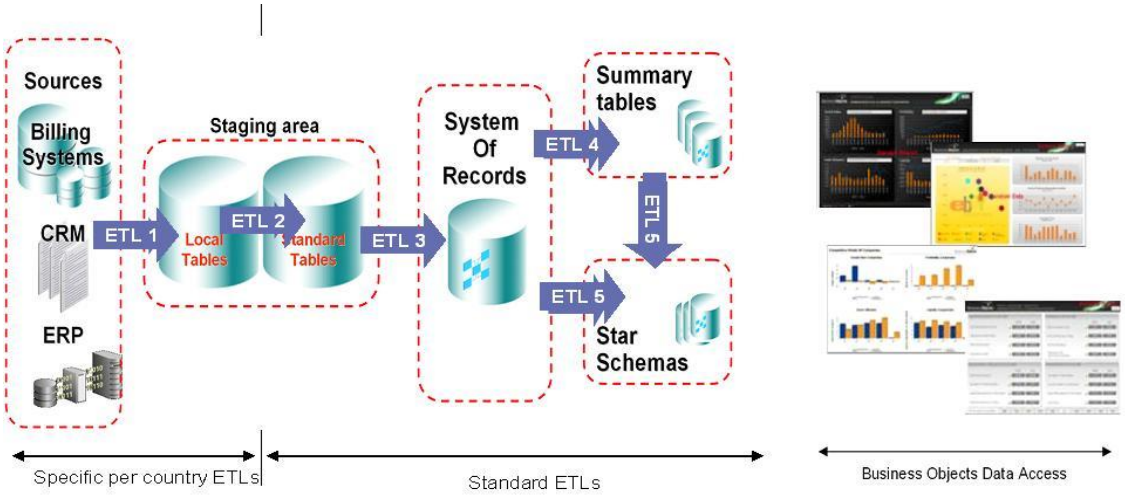
Muchos sistemas tiene la funcionalidad de poder exportar su información en archivos de texto con una estructura definida (por ejemplo separados por comas, ','), o ya más modernos en xml, aquí la dificultad radica en poder tomar estos archivos y cargarlos en el sistema de bases de datos que va a ser nuestro repositorio, la mayoría de las bases de datos poseen herramientas unas más eficientes que otras para este trabajo, ahora bien si nos enfocamos en grandes cantidades de información, digamos unos 200 millones de

registros diarios, estas herramientas de las bases de datos, quedan muy cortas, para estos trabajos tenemos los ETL.

Estas herramientas en muchos casos tienen un costo más elevado que la propia bases de datos, y necesitan más hardware, o son tan versátiles que pueden leer archivos planos con cualquier estructura, ir directamente a la base de datos del sistema origen, etc. son de ambientes gráficos, y se pueden calendarizar, por el hardware que necesitan el tiempo de carga y ejecución lo reducen en algunos casos de 6 horas a 1 hora.

Existe una arquitectura de DWH (*Data Warehouse Architecture* la cual consiste en dividir en dos partes el *Data Warehouse*, una donde está la data ruda donde se transforma y refine llamada *staging area*, el área de presentación es donde se tiene la estructura dimensional y donde el usuario va a acceder la información.

Figura 12. ***Data Warehouse Architecture (with a Staging Area)***



Fuente: elaboración propia.

4.2.2. Almacenamiento de la información

Aquí se debe decidir entre la variedad de bases de datos disponibles la idónea y la que mejor tenga rendimiento en el manejo de millones de registros, según la experiencia *Oracle* se lleva las palmas, SQLServer en sus distintas versiones no le ha llegado a superar, hay en el mercado una nueva base de datos de Sybase IQ que en varias pruebas y que en ciertos aspectos puntuales supera a *Oracle* en rendimiento.

Al elegir la base de datos idónea, es necesario contar con un medio de almacenamiento, se necesita un almacén con suficiente capacidad para guardar Terabytes de información y que esté debidamente soportada por cualquier falla, un RAID 5 sería lo ideal, y que se puedan tener meses y si se puede años guardados.

4.2.3. Análisis de la información

En este punto ya tenemos la información guardada y procesada, ya tenemos en algunos casos resúmenes y en otros la pura data esperando ser analizada, con tanta información se debería de poder tener indicadores que muestren el rumbo de la organización, vamos bien o mal, esto debería ser en tiempo real para corregir el rumbo, pero como no es así, ya que es demasiada la información es necesario un tiempo de análisis, aquí entran otras herramientas que no solo puedan tomar los datos, sino que encontrar relación entre ellos, y así poder mostrar en donde se está fallando y así poder corregir el rumbo, este tipo de herramientas tienen su fuerte en el análisis estadístico y matemático, algunas utilizan algoritmos de inteligencia artificial , podemos mencionar que entre las técnicas de la minería de datos tenemos:

- Redes neuronales
- Árboles de decisión
- Modelos estadísticos
- Agrupamiento o *clustering*

Minería de Datos es hoy lo que muchas organizaciones buscan para poder encontrar en donde está la mina de oro y así salir adelante, la información la tienen sólo falta saber buscar.

4.3. Particiones e índices

4.3.1. Qué es una partición

Los sistemas modernos soportan bases de datos de misiones críticas, manejando varios gigabytes y aún terabytes de información. Uno de los retos es soportar estas grandes bases de datos. Un método es dividirlas en pequeñas partes más manejables, a lo que se le llamamos partición. Dichas particiones también brindan un mejor desempeño, debido a que muchos de los *queries* toman las particiones necesarias para mostrar la información, evitando así leer información innecesaria, lo que produce un mejor manejo de los recursos.

Existen varios tipos de particionamiento que ofrece la base de datos *Oracle*:

- Particionamiento por rango
- Particionamiento por lista de valores
- Particionamiento por función de Hash (Dispersión)
- Particionamiento por rango-lista
- Particionamiento por rango-Hash

Los primeros tres métodos se conocen como particionamiento simple, mientras los últimos dos se denominan métodos de particionamiento compuesto. En el particionamiento simple, sólo se crea un nivel de particiones. El particionamiento compuesto implica la creación de dos niveles de particiones: Particiones y Subparticiones.

4.3.1.1. Particionamiento por rango

El particionamiento por rango, mapea registros a las particiones basándose en rangos de valores de la llave de particionamiento definidos para cada partición. Dicho particionamiento es usualmente utilizado con fechas.

4.3.1.2. Particionamiento por lista de valores

El particionamiento por lista de valores, divide la tabla por valores discretos de una sola de sus columnas. Los valores que definen cada partición, se especifican por medio literales, por ejemplo: ‘amarillo’, ’rojo’, ’verde’ y ’azul’.

4.3.1.3. Particionamiento por función Hash (Dispersión)

Con este particionamiento, los registros se mapean a una partición por medio de la aplicación de una función de dispersión a la llave de particionamiento. El algoritmo utilizado para calcular el valor de dispersión es propio del DBMS y no es modificable, por lo que no se puede determinar a qué partición será asignado un registro dado. Este particionamiento garantiza una distribución equitativa de los registros a través de todas las particiones, pero para lograr tal fin y debido a la naturaleza del mismo, es necesario que el número de particiones sea una potencia de dos, es decir: 2, 4, 8, 16, 32, 128, etc.

4.3.1.4. Particionamiento por rango-lista

En el particionamiento por rango-lista, se crean dos niveles de particionamiento. El primer nivel se basa en rangos de valores, como ocurre en el particionamiento por rango; el segundo nivel se basa en valores discretos, como ocurre en el particionamiento por lista. Esta forma de particionamiento compuesto resulta muy útil para datos

históricos, pero permite además agrupar registros basados en valores de columnas no relacionados o no ordenados.

4.3.1.5. Particionamiento por rango-dispersión

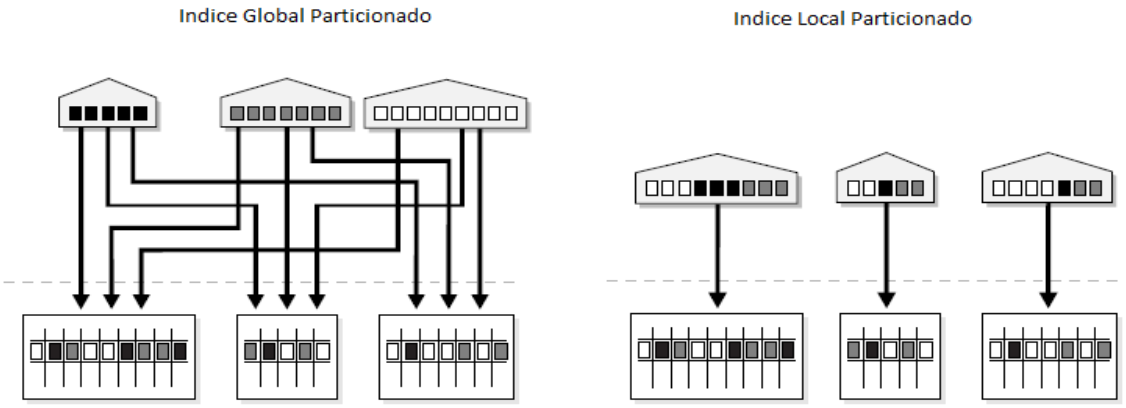
Este método de particionamiento combina las ventajas del crecimiento fijo de las particiones por rango, con la distribución equitativa y el acceso paralelo del particionamiento por dispersión. El primer nivel se basa en rangos de valores, mientras que el segundo nivel se basa el valor de dispersión.

4.3.2. Qué es un índice particionado

Al igual que una tabla particionada, un índice particionado es un índice dividido en segmentos más pequeños y manejables llamados particiones.

Cuando el particionamiento del índice es independiente del particionamiento de la tabla, se le denomina índice particionado global (o índice global). Cuando el índice está ligado al método de particionamiento de la tabla se denomina índice particionado local (o índice local).

Figura 13. **Índices particionados**



Fuente: elaboración propia.

4.3.2.1. Índices particionados globales

Un índice particionado de tipo global, es aquel cuyas particiones no están asociadas a una partición correspondiente dentro de la tabla para la cual se crea. Este tipo de índice puede crearse tanto para tablas particionadas como para tablas no particionadas.

4.3.2.2. Índices particionados locales

Un índice particionado de tipo local sólo puede crearse para tablas particionadas. Cada partición del índice está asociada a una partición correspondiente dentro de la tabla, es decir, cada partición del índice se trata como un índice de su partición de tabla correspondiente

CONCLUSIONES

1. Se logró cargar satisfactoriamente, la información académica hasta el cierre del ciclo académico 2009.

2. Elaboración de una guía detallada para la correcta actualización del *DataMart*, para que éste se mantenga actualizado y que siempre sea de apoyo para la correcta toma de decisiones.

3. *Oracle*® y *Oracle Discoverer*® son herramientas que bien aplicadas, permiten excelentes resultados dentro de cualquier organización.

4. No existe un procedimiento establecido y documentado para ejecutar o detener la aplicación por completo. Por lo que al reiniciar el servidor, la aplicación hay que ponerla en funcionamiento manualmente.

5. Cuando se recabó la información para acceder al servidor no existía un documento de cómo hacerlo, ni una persona encargada del mismo, por consiguiente el equipo se encontraba en desuso.

6. El uso de índices con tablas particionadas ayuda a tener fácil acceso a datos con grandes volúmenes de información, optimizando espacio y tiempo.

RECOMENDACIONES

1. Creación de un proceso administrativo, en el cual se establezcan los siguientes pasos:

   a) Establecer un tiempo de entrega límite de las notas en cada cierre de ciclo.

   b) Generar la información del sistema central de Control Académico de forma automática y trasladarla hacia el servidor de *Data Warehouse*, con al menos una semana de retraso, para que éste sirva para la verificación de las notas ingresadas.

   c) Procesar la información, siguiendo la guía que este trabajo contiene.

2. Creación de un proceso de auditoría de sistemas, con la finalidad de poder controlar y verificar los accesos a la información a este servidor.

3. Una vez creado el proceso recomendado en el numeral uno de estas recomendaciones, es factible iniciar a incorporar la información académica de las demás escuelas de la Facultad de Ingeniería, ya que el sistema fue diseñado para este propósito.

4. Creación de un portal de indicadores gerencial el cual muestre una visión global de la situación actual sobre las notas de estudiantes, capacidad de salones, demanda de cursos, etc.

5. Se recomienda la creación de una política de respaldo tanto para la base de datos de la aplicación, así como al sistema operativo del servidor.

BIBLIOGRAFÍA

1. ABBEY, Michael. *Oracle Data Warehousing.* San Fco. CA: McGraw-Hill, 2008. 685 p. ISBN: 0-07-882511-3.

2. NAIR, Pratap. *Large amount of data requires an outside management system* [en línea]. Houston, TX, Marzo 2007 [ref. de 20 de marzo 2009]. Disponible en Web: <http://houston.bizjournals.com/houston/stories/2007/3/9/focus18.html>. [Consulta 20 de marzo de 2009].

3. ORACLE. *Oracle® Business Intelligence Discoverer* [en línea]. Redwood City, CA, Julio 2005 [ref. de 15 de marzo 2009]. Disponible en Web: <http://download.oracle.com/docs/pdf/B13916_04.pdf>. [Consulta 15 de marzo de 2009].

4. REYES MARROQUÍN, Mario Roberto; ROSALES TEJADA Pablo Augusto. "Desarrollo de un *DataMart* de información académica de estudiantes de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería Universidad de San Carlos de Guatemala". Trabajo de graduación de Ing. en Ciencias y Sistemas, Facultad de Ingeniería. Universidad de San Carlos de Guatemala, 2007. 124 p.

5. WIKIPEDIA. *Extrac, transform and load* [en línea]. Julio 2009 [ref. de 15 de abril 2009]. Disponible en Web: <http://es.wikipedia.org/wiki/Extract,_transform_and_load>. [Consulta 15 de abril de 2009].

6. _____ . *Minería de datos* [en línea]. Julio 2011 [ref. de 21 de marzo 2009]. Disponible en Web: <http://es.wikipedia.org/wiki/Miner%C3%ADa_de_datos>. [Consulta 22 de julio de 2011].

APÉNDICE

1. ANTECEDENTES DEL PROYECTO

1.1. Apuntes teóricos

1.1.1. Qué es un ETL

Entendemos ETL (son las siglas del inglés de *Extract, Transform y Load*), como el proceso de extracción, transformación y carga de los datos, que es parte del ciclo de vida de una implementación de *Data Warehouse*. La forma en que funciona es utilizando una herramienta de ETL, se conecta a la fuente de los datos, se hace la transformación dentro de la misma herramienta y carga los datos a la base de datos destino.

Proceso de extracción

Es el primer paso de obtener información desde los sistemas transaccionales (también llamados *legacy system*) hacia el ambiente del *Data Warehouse*. Uno de los retos de esta etapa, es la de degradar lo menos posible el sistema origen, ya que si el proceso de extracción demanda mucho recurso de éste, podría colapsarlo.

Proceso de transformación

Una vez que la información es extraída hacia el área de tráfico de datos, hay posibles pasos de transformación como: Limpieza de la información (*data Cleaning*), seleccionar únicamente los campos necesarios, combinar fuentes de datos, traducir

códigos, obtener nuevos valores calculados, unir datos de múltiples fuentes, etc. La validación de los datos, podría dar como resultado Error con lo que debemos ejecutar una política de manejo de excepciones (rechazar el registro completo, dar al campo un valor nulo, etc.).

Proceso de carga

Una vez terminado el proceso de Transformación, se procede entonces a cargar la información en el sistema destino. Para ello existen dos técnicas:

1. Acumulación Simple: es la más sencilla y consiste en sumarizar todas las transacciones comprendidas, en una única transacción de Suma o Promedio en el *Data Warehouse*
2. *Rolling*: Este proceso aplica para aquellos casos en los que se requieren múltiples niveles de granularidad. Para ello se almacena la información resumida a distintos niveles jerárquicos.

Un desarrollo recientemente de los sistemas de ETL ha hecho posible el procesamiento paralelo. Esto ha permitido desarrollar varios métodos para mejorar el desempeño, principalmente con el manejo de grandes volúmenes de datos. También se han incorporado procesos para la depuración y control de calidad para los datos, como por ejemplo *Data Profiling, Data Cleansing y Data Auditing*.

Data Profiling

Es el proceso de examinar los datos que existen en las fuentes de origen de una organización y recopilar estadísticas e información sobre los mismos. El propósito de dichas estadísticas es:

- Determinar qué datos pueden ser usados para otros propósitos.

- Conseguir métricas de calidad de datos que incluyen si los datos cumplen los estándares de la organización.
- Reduce el riesgo de integrar información a nuevas aplicaciones dado que conocemos su estado.
- Permite hacer un seguimiento de la calidad de datos.
- Capacidad de entender problemas derivados de los datos en proyectos que hagan uso intensivo de los mismos.
- Tener una visión global de los datos de la organización para desplegar políticas de *Data Governance.*

Data Cleansing

Es el proceso de detectar o descubrir y corregir datos corruptos, incoherentes o erróneos de un conjunto de datos. Después del proceso la información será consistente con otros conjuntos similares de datos. La validación de datos puede ser estricta o mediante el uso de la lógica Fuzzy.

Este proceso permite detectar entradas duplicadas, incompletas,… y establecer reglas para corregirlas. El objetivo no es borrar información como tal, sino mejorar la calidad de los datos construyendo un proceso de mejora continua.

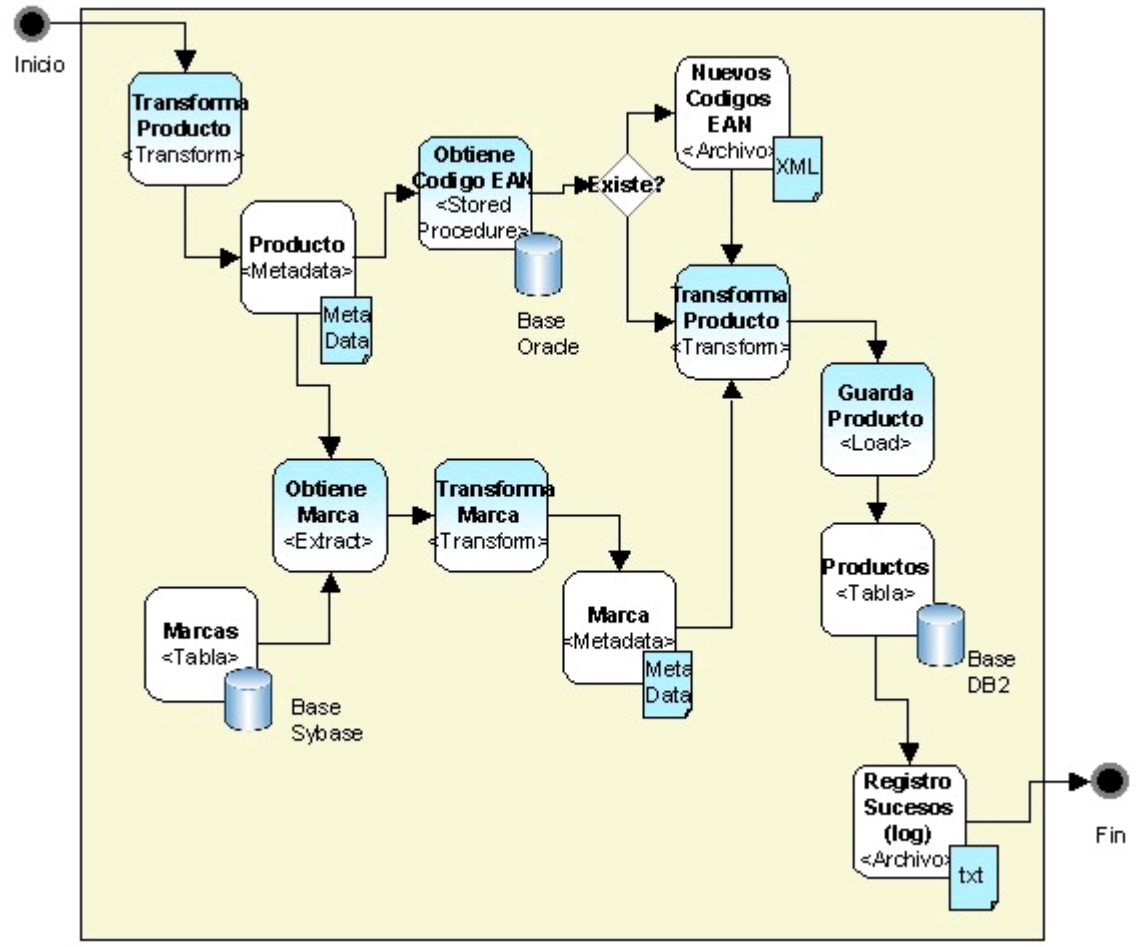
Data Auditing

Es el proceso de gestionar cómo los datos se ajustan a los propósitos definidos por la organización. Se establecen políticas para gestionar los criterios de datos para la organización. No es suficiente con actuar sino que se debe vigilar.

En el mercado del software, encontramos varias herramientas para hacer ETL, algunas requieren de compra de licencia, mientras que otras son de software libre:

- IBM DataStage
- *Oracle Warehouse Builder*
- *Data Integrator de Business Objects*
- *Cognos Decisionstream*
- *Microsoft Data Integration Services*
- *Talend Open Studio* (libre)
- Kettle (libre)
- Scriptella (libre)

Ejemplo



Fuente: elaboración propia.

A este proceso ejemplo se le entrega un producto, el cual lo transforma, para obtener la marca asociada (desde una base Sybase), y por otro lado, su código EAN (usando un *Stored Procedure*), si el código EAN no existe se busca en un archivo XML, luego finalmente toda la información se carga en la tablas de Productos (Base DB2), y además se registra en un archivo de Log (archivo txt). Más que nada este ejemplo sirve para mostrar la relación que se puede establecer entre los distintos objetos ETL, pero la estructura definitiva de un diseño ETL dependerá de la herramienta que se utilice.

1.1.2. *SQL*Loader*

SQL*Loader es una potente utilidad *Oracle*® de importación de datos que hace posible la carga automática de datos externos (residentes en archivos del sistema operativo) en tablas de la base de datos *Oracle*. Los datos pueden cargarse en una o varias tablas que previamente deben estar creadas y que pueden o no tener contenido previo. Los nuevos datos podrán sustituir a los que ya existieran en las tablas o bien añadirse como nuevas filas.

Es posible la carga de datos almacenados en forma de archivos de texto (lo más corriente) o binarios. La entrada del programa consiste en:

- Uno o varios archivos de datos, con nombres por defecto terminados en .DAT, conteniendo los datos a importar.
- Un archivo (texto) de control, con nombre por defecto terminado en .CTL, que contiene órdenes que permiten guiar y particularizar el proceso de carga de datos. En este archivo se especifican los atributos de las tablas de la base de datos en los que se van a insertar los valores contenidos en el archivo de datos.

Como salida se generan hasta tres archivos de texto:

- Un archivo de resultados, con nombre por defecto terminado en .LOG, que contiene diversos informes sobre la realización del proceso de carga de datos.
- Un archivo de errores, con nombre por defecto terminado en .BAD, que contiene aquellos datos del archivo de entrada que no han podido ser cargados en la base de datos por diversos errores. Si la carga se ha realizado sin errores entonces no se genera este archivo.
- Un archivo de descartados, con nombre por defecto terminado en .DSC, que contiene aquellos datos del archivo de entrada que no han sido cargados en la base de datos porque así se había especificado en el archivo de control. Si no se ha descartado ningún dato durante el proceso de carga no se genera este archivo.

La diferencia entre los datos erróneos y los descartados es que los primeros son aquellos datos que no han sido insertados en las tablas de la base de datos porque no han podido ser leídos correctamente desde el archivo de datos de entrada o bien su inserción causa errores de incumplimiento de restricciones definidas en las tablas, mientras que los datos descartados no se insertan en la base de datos porque no verifican una determinada condición que puede imponerse en el archivo de control, de manera que tan solo se inserten los datos que satisfacen dicha condición.

Un concepto importante a la hora de trabajar con *SQL*Loader* es el de registro físico y registro lógico. Suponiendo que el archivo de datos es de texto, un registro físico es una línea del archivo. Un registro lógico se corresponde con una fila o tupla de una tabla. La equivalencia entre estos dos tipos de registros puede ser cualquiera. Un registro físico puede dar lugar a varios registros lógicos de manera que cada lectura de una línea del archivo de entrada produce la escritura de varias filas en las tablas. Si por el contrario se necesitan varios registros físicos para formar un registro lógico, se necesitará leer varias líneas del archivo de datos antes de proceder a la inserción de una fila. El caso más sencillo y conveniente es cuando se produce la identidad entre ambos tipos de registro, en el cual cada lectura de una línea del archivo de datos puede generar una inserción en la base de datos.

Mediante la adecuada especificación del archivo de control puede programarse en cierta medida el proceso de carga de datos ya que se dispone de una cierta capacidad de especificación de condiciones para determinar si los datos leídos han de insertarse o no en la base de datos. La estructura del archivo de datos de entrada puede ser fija o secuencial. Es fija si los valores están dispuestos de forma invariable en posiciones determinadas del archivo de datos y secuencial si simplemente los valores están colocados uno a continuación del otro en cada línea del archivo y separados por un símbolo determinado. En la práctica pueden existir archivos cuya estructura presente características de los dos tipos.

No es obligatorio que los valores de todos los atributos o columnas del registro lógico aparezcan especificados en los campos del registro físico; se pueden leer todos o tan solo parte de los atributos. Los atributos cuyos valores no aparecen en el archivo de datos se insertan en las tablas con valores nulos.

Aunque el archivo de control permite una amplia gama de posibilidades para particularizar la carga de datos a las características del archivo de entrada, puede ser necesario o conveniente preprocesar los datos (por ejemplo, en un editor de texto) eliminando datos innecesarios, contenidos redundantes, etc. Si los datos a ser cargados están ordenados según algún atributo o combinación de atributos entonces se puede aprovechar esta circunstancia para acelerar el proceso.

Es muy interesante la posibilidad de insertar valores a determinados atributos de las tablas automáticamente, sin necesidad de especificar dichos valores en el archivo de datos. Por ejemplo, se puede tener un atributo en una tabla cuyo valor se vaya incrementando en una cantidad fija cada vez que se inserta una fila en esa tabla, o bien el valor a insertar para ese atributo es constante para cada fila insertada.

Para ejecutar *SQL*Loader* se necesita especificar el nombre y palabra clave en el sistema *Oracle* de un usuario (usualmente el propietario) que disponga de permiso de

inserción (INSERT) sobre las tablas en las que se van a cargar los datos. En la invocación del programa se ha de especificar el archivo de datos a procesar y el archivo de control, de la siguiente manera:

sqlloader usuario/palabra_clave CONTROL=archivo_control DATA=archivo_datos

Carga de los datos

En primer lugar se define en SQL*Plus una tabla en la cual insertar la información contenida en el archivo de datos Poblacion.dat.

CREATE TABLE población(
num NUMBER(3) PRIMARY KEY,
nombre CHAR(30) NOT NULL,
hombres NUMBER(6) NOT NULL,
mujeres NUMBER(6) NOT NULL);

En esta tabla se almacena el nombre de cada municipio y su población masculina y femenina junto con un número de orden secuencial para cada municipio.

En este caso existe una identidad clara uno a uno entre registro físico y registro lógico. Veamos la forma de obtener el valor a insertar para cada fila de la tabla.

Atributo	**Valor**
Num	Valor generado automáticamente, con incremento unidad, para cada línea del archivo de entrada.
Nombre	Especificado en las columnas 7 a 35 de cada línea del archivo de datos.
hombres	Especificado en las columnas 44 a 49 de cada línea del archivo de datos.
Mujeres	Especificado en las columnas 51 a 56 de cada línea del archivo de datos.

Fuente: elaboración propia.

Obsérvese que el atributo num del registro lógico no figura en el registro físico, mientras que el dato de población total del municipio, que sí consta en el registro físico, no se ha incluido como atributo de la tabla (no aparece en el registro lógico) por poder calcularse fácilmente como la suma de la población masculina y femenina.

La redacción del archivo de control requiere utilizar un editor de texto, por ejemplo emacs. En nuestro caso se redactará el archivo carga.ctl con el siguiente contenido:

```
LOAD DATA
APPEND
INTO TABLE poblacion(
num SEQUENCE(COUNT,1),
nombre POSITION(07:35) CHAR,
hombres POSITION(44:49) INTEGER EXTERNAL,
mujeres POSITION(51:56) INTEGER EXTERNAL)
```

Mediante *APPEND* se indica que los datos procedentes del archivo de entrada se añadirán a los que ya existieran en la tabla. La cláusula *INTO TABLE* especifica el nombre de la tabla en la que se insertarán los datos leídos. Como el formato del archivo de datos es claramente fijo, se especifica para cada atributo de la tabla a leer las posiciones de la línea del archivo de datos en que se encuentra. Para el atributo num se genera automáticamente un número, con valor inicial el número de filas que ya estuvieran almacenadas en la tabla (si la tabla está vacía) que se incrementa en uno cada vez que se inserta una nueva fila.

Los tipos de datos especificados en el archivo de control se refieren a los datos almacenados en el archivo de datos y no a los atributos correspondientes de la tabla donde se van a cargar.

Una vez que ya se dispone del archivo de datos, del archivo de control y de las tablas donde se van a insertar los datos puede iniciarse el proceso de carga datos mediante la invocación del programa *SQL*Loader*:

sqlloader usuario/palabra_clave CONTROL=carga.ctl, DATA=Poblacion.dat

La ejecución genera únicamente el archivo carga.log. No se ha generado un archivo con la relación de datos de entrada erróneos, pues no se ha producido ningún error, ni tampoco se ha descartado ningún dato del archivo de datos ya que no se ha especificado ninguna condición a tal efecto, en el archivo de control (cláusulas *WHEN*). Todo esto puede comprobarse examinando los diversos contenidos del archivo de resultados.

105 Rows successfully loaded.
0 Rows not loaded due to data errors.
0 Rows not loaded because all WHEN clauses were failed.
0 Rows not loaded because all fields were null.

La información más interesante especificada en el archivo de resultados es el número total de filas insertadas en la base de datos. Puede comprobarse que la información fue realmente insertada en la tabla realizando una consulta desde SQL*Plus.

SELECT * FROM población;

1.1.3. PL/SQL

Es lenguaje de programación embebido en *Oracle* y PostgreSQL.

El PL/SQL soporta todas las consultas y manipulación de datos que se usan en SQL, pero incluye nuevas características:

- El manejo de variables.
- Estructuras modulares.
- Estructuras de control de flujo y toma de decisiones.
- Control de excepciones.

El lenguaje PL/SQL está incorporado en:

- Servidor de la base de datos.
- Herramientas de *Oracle* (*Forms, Reports, Discoverer*, ...).

En un entorno de base de datos los programadores pueden construir bloques PL/SQL para utilizarlos como procedimientos o funciones, o bien pueden escribir estos bloques como parte de scripts SQL*Plus.

Los programas o paquetes de PL/SQL se pueden almacenar en la base de datos como otro objeto, y todos los usuarios que estén autorizados tienen acceso a estos paquetes. Los programas se ejecutan en el servidor para ahorrar recursos a los clientes.

A continuación se muestra un listado de los tipos de datos disponibles en *Oracle* y PLSQL:

Tipo de dato / Sintaxis	**Descripción**
dec(p, e)	Donde “p” es la precisión y “e” la escala. Por ejemplo: dec(3,1) es un número que tiene 2 dígitos antes del decimal y un dígito después del decimal.
decimal(p, e)	Donde “p” es la precisión y “e” la escala. Por ejemplo: decimal(3,1) es un número que tiene 2 dígitos antes del decimal y un dígito después del decimal.
Doble precisión	
Flotante	
Int	
Integer	
numeric(p, e)	Donde “p” es la precisión y “e” la escala. Por ejemplo: numeric(7,2) es un número que tiene 5 dígitos antes del decimal y 2 dígitos después del decimal.
number(p, e)	Donde “p” es la precisión y “e” la escala. Por ejemplo: number(7,2) es un número que tiene 5 dígitos antes del decimal y 2 dígitos después del decimal.
Real	
Smallint	
char (tamaño)	Donde tamaño es el número de caracteres a almacenar. Son cadenas de ancho fijo. Se rellena con espacios.

	Continua
varchar2 (tamaño)	Donde tamaño es el número de caracteres a almacenar. Son cadenas de ancho variable.
Long	Son cadenas de ancho variable.
Raw	Son cadenas binarias de ancho variable.
long raw	Son cadenas binarias de ancho variable.
Date	
timestamp (fractional seconds precision)	Incluye año, mes día, hora, minutos y segundos. Por ejemplo: timestamp(6)
timestamp (fractional seconds precision) with time zone	Incluye año, mes día, hora, minutos y segundos; con un valor de desplazamiento de zona horaria. Por ejemplo: timestamp(5) with time zone
timestamp (fractional seconds precision) with local time zone	Incluye año, mes día, hora, minutos y segundos; con una zona horaria expresada como la zona horaria actual. Por ejemplo: timestamp(4) with local time zone
interval year (year precision) to month	Período de tiempo almacenado en años y meses. Por ejemplo: interval year(4) to month
urowid [tamaño]	Rowid universal. Donde tamaño es opcional.

	Continua
Boolean	
nchar (tamaño)	Donde tamaño es el número de caracteres a almacenar. Cadena NLS de ancho fijo.
continua nvarchar2 (tamaño)	Donde tamaño es el número de caracteres a almacenar. Cadena NLS de ancho variable.
Bfile	Localizadores de archivo apuntan a un objeto binario de sólo lectura fuera de la base de datos.
Blob	Localizadores LOB apuntan a un gran objeto binario dentro de la base de datos.
Clob	Localizadores LOB apuntan a un gran objeto de caracteres dentro de la base de datos.
Nclob	Localizadores LOB apuntan a un gran objeto NLS de caracteres dentro de la base de datos.

Fuente: elaboración propia.

Identificadores

Un identificador es un nombre que se le pone a un objeto que interviene en un programa, que puede ser variable, constante, procedimientos, excepciones, cursores... Debe tener un máximo de 30 caracteres que empiece siempre por una letra, y puede contener letras, números, los símbolos $, #, _, y mayúsculas y minúsculas indiferentemente. Los identificadores no pueden ser palabras reservadas (SELECT, INSERT, DELETE, UPDATE, DROP).

Operador de asignación	:= (dos puntos + igual)
Operadores aritméticos	+ (suma) - (resta) / (división) ** (exponente)
Operadores relacionales o de comparación	= (igual a) <>, != (distinto de) < (menor que) > (mayor que) >= (mayor o igual a) <= (menor o igual a)
Operador de concatenación	||
Comentarios	/* comentario de dos o más líneas */ -- comentario de una línea

Fuente: elaboración propia.

Variables

Las variables son nombres para procesar los elementos de los datos. Declaración:

```
Nombre_variable tipo [NOT NULL] [:= valor | DEFAULT valor]
```

:= y **DEFAULT** son lo mismo. Si ponemos **NOT NULL** es obligatorio inicializar la variable.

Ejemplos:

```
num_dep NUMBER(2) NOT NULL :=20
```

```
num_emple VARCHAR2(15) DEFAULT 'Pedro'
```

También se puede definir una variable a partir de un campo mediante los atributos **%TYPE** y **%ROWTYPE**, con esto damos el tipo y longitud a la variable de otra variable u objeto ya definido.

%TYPE es la que se utiliza normalmente, **%ROWTYPE** es para claves de registro. El **NOT NULL** y el valor inicial no se heredan, sólo el tipo de dato y longitud de ese dato.

Por ejemplo:

```
num_dep emple.dept_no%TYPE
```

Constantes

Las constantes son como las variables pero no puede modificarse su valor. Se declaran de la siguiente manera:

```
nombre_constante CONSTANT tipo_de_dato := valor
```

Por ejemplo, el IVA es un valor fijo, y para declararlo lo haríamos de la siguiente manera:

```
Imp_iva constant number(2,2) := 12,5
```

Bloque PL/SQL

Bloque es la unidad de estructura básica en los programas PL/SQL. Supone una mejora en el rendimiento, pues se envían los bloques completos al servidor para ser procesados en lugar de enviar cada secuencia SQL.

Partes de un bloque:

- Zona de declaraciones: zona opcional. Se declaran los objetos locales (variables, constantes...).
- Zona de instrucciones: zona obligatoria.
- Zona de tratamiento de excepciones: zona opcional. Se tratan excepciones en el programa.

Forma de crear un bloque:

```
[ DECLARE   |   IS / AS ]
        <declaraciones>
BEGIN
        <instrucciones>
[ EXCEPTION ]
        <tratamiento de excepciones>
END;
/
```

La barra "/" siempre se pone al final para ejecutar el bloque.

Tipos de bloques

- Anónimo (sin nombre)

  Siempre comienza con **DECLARE** o directamente con **BEGIN**.

  Ejemplo 1:

```
BEGIN
  DBMS_OUTPUT.PUT_LINE ('Hola');
END;
/
```

DBMS_OUTPUT es un depurador de *Oracle* que sirve para visualizar cualquier cosa, pero antes lo debemos tener activado:

```
SET SERVEROUTPUT ON;
```

Ejemplo 2:

```
DECLARE
        v_precio number;
BEGIN
        select pvp into v_precio
        from tarticulos
        where codigo=100;
        dbms_output.put_line (v_precio);
END;
/
```

Ejemplo 3:

El siguiente bloque anónimo nos muestra la fecha actual con el formato "Martes, 18 de marzo de 1998, a las 13:04:55".

```
DECLARE
        fecha date;
BEGIN
        select sysdate into fecha from dual;
        dbms_output.put_line (to_char(sysdate,
        'day", "dd" de "month" de "yyyy", a las "hh24:mi:ss'));
END;
/
```

- Subprogramas (tienen nombre)

Se pueden almacenar en la base de datos.

Existen dos tipos de subprogramas:

Procedimientos en PLSQL

Los procedimientos tienen la utilidad de fomentar la reutilización de programas que se usan comúnmente. Una vez compilado, queda almacenado en la base de datos y puede ser utilizado por múltiples aplicaciones.

La sintaxis es la siguiente

```
CREATE [OR REPLACE] PROCEDURE nombre_procedimiento
 [nombre_parametro modo tipodatos_parametro  ]
IS | AS
  bloque de código
```

Donde "modo" puede contener los valores *IN, OUT, IN OUT*. Por defecto tiene el valor *IN* si no se pone nada. *IN* indica que el parámetro es de entrada y no se podrá modificar. *OUT* indica que el parámetro es de salida con lo que el procedimiento devolverá un valor en él. *IN OUT* indica que el parámetro es de entrada/salida. Con lo que al llamar al procedimiento se le dará un valor que luego podrá ser modificado por el procedimiento y devolver este nuevo valor.

"tipodatos_parametro indica el tipo de datos que tendrá el parámetro según lo indicado en Tipos de datos *Oracle*/PLSQL

Para borrar un procedimiento almacenado de la base de datos

```
DROP PROCEDURE nombre_procedimiento
```

Para utilizar un procedimiento almacenado de la base de datos

Simplemente se llama desde un bloque anónimo (desde la línea de comandos), previamente habiendo inicializado el/los parámetro/s (en caso que existan).

```
DECLARE
  nombre_parametro tipodatos_parametro;
BEGIN
  nombre_parametro tipodatos_parametro := valor_de_inicializacion;
  nombre_procedimiento (nombre_parametro => nombre_parametro);
END;
/
```

Funciones en PLSQL

Una función es un bloque de código PL/SQL que tiene las mismas características que un procedimiento almacenado. La diferencia estriba que una función devuelve un valor al retornar. Al devolver un valor puede ser llamado como parte de una expresión.

La sintaxis sería la siguiente:

```
CREATE [OR REPLACE] FUNCTION nombre_función
  [nombre_parámetro modo tipodatos_parametro  ]
RETURN  tipodatos_retorno  IS | AS
   bloque de código
```

Donde "modo" puede contener los valores IN, OUT, IN OUT. Por defecto tiene el valor IN si no se pone nada. IN indica que el parámetro es de entrada y no se podrá modificar. OUT indica que el parámetro es de salida con lo que el procedimiento devolverá un valor en él. IN OUT indica que el parámetro es de entrada/salida. Con lo que al llamar al procedimiento se le dará un valor que luego podrá ser modificado por el procedimiento y devolver este nuevo valor. Sin embargo, en este caso sólo tendría sentido(por el concepto de función en sí mismo) declarar parámetros del tipo IN y devolver el valor como retorno de la función.

"tipodatos_parametro" y "tipodatos_retorno" indican el tipo de datos que tendrá el parámetro y el valor de retorno de la función respectivamente según lo indicado en Tipos de datos *Oracle*/PLSQL

Para borrar una función de la base de datos

```
DROP FUNCTION nombre_función
```

Los procedimientos y funciones se pueden agrupar en unas estructuras llamadas Paquetes

Triggers

Se ejecuta ante un determinado evento de manera automática. Generalmente se utilizan para garantizar que una determinada acción siempre se realiza después de realizar una tarea determinada. Se debe tener cuidado con este tipo de estructuras ya que su uso excesivo puede dar lugar a dependencias difíciles de mantener. Además, se deben tener muy claros las restricciones de integridad, para evitar problemas.

La sintaxis sería la siguiente:

A nivel de sentencia:

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
 momento_ejecución evento [evento] ON nombre_tabla
   bloque PLSQL;
```

A nivel de registro:

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
 momento_ejecución evento [evento] ON nombre_tabla
[REFERENCING OLD AS old | NEW AS new]
FOR EACH ROW
```

```
[WHEN condición]
   bloque PLSQL;
```

Donde "momento_ejecución" indica cuando se ejecuta el *trigger* automáticamente. Puede contener los valores BEFORE ó AFTER.
"evento" indica la operación que provoca la ejecución de este bloque. Puede contener los valores :*INSERT, UPDATE DELETE*.
"old" indica el nombre que se le da al registro con los valores antiguos que se tenían antes de :la ejecución de la operación que activó el *trigger*. Mientras que "*new*" indica el valor que tiene actualmente después de dicha operación.
Con la cláusula "*WHEN*" se puede indicar una restricción que haga que el *trigger* se ejecute o no. :Por ejemplo se puede indicar que el *trigger* se ejecute solo si el campo "campo1" de la tabla tiene :un valor mayor que 50.

La cláusula "FOR EACH ROW" indica que el trigger es a nivel de registro.

Para eliminar un trigger:

```
DROP TRIGGER nombre_trigger
```

1.1.4. Shell Scripts:

Los shell scripts son programas escritos con comandos UNIX y son equivalentes a los batch de DOS aunque más potentes, admiten ejecuciones en segundo plano (*background*) y tienen un conjunto de expresiones más amplio.

Para realizar un shell script en condiciones, al igual que para realizar un programa en cualquier lenguaje de programación, es necesario tener claro lo que se quiere obtener y de qué situación se parte. Es mejor perder un poco de tiempo en realizar un desarrollo mental del programa que dedicarse a teclear y probar cosas, lo más seguro

es que no se consiga el objetivo, y si se consigue la estructura del programa no será la adecuada.

Una de las ventajas que presentan los *shell scripts* es que pueden ser portadas de una máquina UNIX a otra sin problemas, sin necesidad de retocar nada, salvo que se utilicen llamadas a programas muy concretos específicos de una versión de UNIX, mientras que los programas compilados (desarrollados en C, Pascal, etc.) deben ser recompilados, pues el código se generará en función del microprocesador de cada máquina. Otra ventaja es la facilidad de lectura e interpretación.

El principal inconveniente que presentan respecto a los programas compilados es la lentitud de ejecución, que se puede paliar usando las utilidades *built-in* incluidas en el propio *kernel* en lugar de llamar a comandos externos que han de ser leídos de disco. Otro inconveniente es que el código resulta visible a cualquier usuario que lo pueda ejecutar.

Se deben añadir comentarios con el fin de facilitar la lectura del programa. Los comentarios se insertan anteponiendo el carácter "**#**" al comentario, que se extenderá hasta el final de la línea.

Los *scripts* suelen encabezarse con comentarios que indican el nombre de archivo y lo que hace el script. Se colocan comentarios de documentación en diferentes partes del script para mejorar la comprensión y facilitar el mantenimiento. Un caso especial es el uso de "**#**" en la primera línea, seguido del carácter admiración y el *path* de la subshell, para indicar el intérprete con que se ejecutará el script:

#!/bin/ksh

Es interesante saber que muchos comandos devuelven un valor después de ejecutarse, y que este valor indicará si la ejecución ha sido buena o si ha habido algún fallo y qué tipo de fallo se ha producido. Para conocer si un comando devuelve o no un

valor y qué es lo que devuelve en cada caso se deberá consultar la documentación, pero por lo general en caso de una ejecución correcta devolverán el valor 0, y en caso de fallo otro número, positivo o negativo.

Para ejecutar un archivo de comandos es necesario que tenga activados, al menos, los permisos de lectura y ejecución.

Variables:

Para asignar un valor a una variable:

valor=7

Para visualizar el contenido de la variable

root@ubuntu:~# echo $valor 7

Para borrar el contenido

root@ubuntu:~# unset valor

root@ubuntu:~# echo $valor

A continuación se muestra una tabla con una serie de variables especiales que serán de utilidad a la hora de escribir nuestros scripts:

VARIABLE	DESCRIPCIÓN
$0	Nombre del Shell-Script que se está ejecutando.
$n	Parámetro o argumento pasado al Shell-Script en la posición n, n=1,2,...
$PS1	Prompt
$#	Número de argumentos.
$*	Lista de todos los argumentos.

$?	Salida del último proceso ejecutado.
$$	Número de identificación del proceso (PID)
$!	Número del último proceso invocado por la Shell

Fuente: elaboración propia.

Variables de entorno

Existen dos áreas de memoria en las *shells* para almacenar variables, el Área local de datos y el Entorno. Podemos visualizar su contenido de ambas áreas, al igual que Windows, con el comando set.

Por defecto, cuando se asigna un valor a una variable, es local, es decir, es conocida por esa *shell*, pero si se abre otra *shell* a partir de la que estamos, estas nuevas '*subshells*' desconocen el valor de las variables que hemos asignado anteriormente.

En cambio, las variables del entorno están disponibles para las *subshells*. Es decir los valores de estas variables son conocidos por los procesos hijos de la *shell*.

Para hacer que una variable se almacene en el área de entorno, se utiliza el siguiente comando:

root@ubuntu:~# export nombre=javi

Para ver la lista de variables del entorno usaremos el comando env.

Se debe saber que una variable exportada NO es lo mismo que una variable global, sino una copia, ya que podrá ser modificada pero volverá a tener su anterior valor cuando salga de la subshell.

Cómo ya sabemos, existe un conjunto de variables de entorno predefinidas, que a continuación listamos en esta tabla:

COMANDO	DESCRIPCIÓN
PATH	Camino de búsqueda de órdenes
HOME	Directorio de trabajo del usuario
USER	Usuario que estableció la sesión
PWD	Ruta completa del directorio de trabajo actual
LOGNAME	Nombre del usuario que ejecuta la Shell
TERM	Tipo de terminal.
SHELL	Shell que está ejecutándose
PS1, PS2, PS3, PS4	Prompts

Fuente: elaboración propia.

Entrecomillado

Debido a que la *shell bash* puede usar nombres simbólicos que representan valores, como el *path* del directorio personal, necesitaremos conocer la diferencia entre los distintos tipos de entrecomillado, el simple, el doble y el uso de las secuencias de escape ().

Entrecomillado simple

Sirve para hacer que la *shell* tome lo encerrado entre él como una expresión literal, ya sean variables, comodines u otras expresiones entrecomilladas.

Es decir el contenido no es interpretado por la shell.

Para entenderlo mejor, imaginemos que deseamos crear una carpeta de nombre * (asterisco).

Si lo intentamos, es decir si tecleamos mkdir *, dará error. Sin embargo, si escribimos mkdir '*', el intérprete de comandos tomará el carácter comodín (*) como un literal y no como un comodín, y podremos crear la carpeta con ese nombre.

Entrecomillado doble

Permiten la sustitución de parámetros, la sustitución de comandos y la evaluación de expresiones aritméticas, pero ignoran los caracteres tubería, sustituciones de tilde, expansión de caracteres comodines, aliases y la división de palabras vía delimitadores.

Las comillas simples dentro de las comillas dobles no tienen efecto. Se puede incluir comillas dobles dentro de una cadena con comillas dobles anteponiendo.

Es decir, se produce sustitución de variable (el signo del dólar se interpreta) por ejemplo, podremos ejecutar ls "$BASH" y nos devolverá correctamente el tipo de shell (/bin/bash), pero si hacemos ls "*", el comodín será tomado como un literal.

Secuencias de escape

Una barra inclinada inversa no entrecomillada, es decir, preserva el valor literal del siguiente carácter que lo acompaña.

Para entenderlo mejor, veamos un ejemplo:

root@ubuntu:~# echo $BASH

/bin/bash

Si utilizamos las secuencias de escape:

root@ubuntu:~# echo $BASH

$BASH

Veamos otros ejemplos:

root@ubuntu:~# usuario=javi

root@ubuntu:~# echo $usuario

javi

root@ubuntu:~# echo "$usuario"

javi

root@ubuntu:~# echo '$usuario'

$usuario

root@ubuntu:~# echo $usuario

$usuario

root@ubuntu:~# echo "'$usuario'"

'javi'

A continuación se muestra una tabla para que fácilmente veamos la diferencia entre un tipo de entrecomillado y otro:

EXPRESIÓN	VALOR
$usuario	javi
"$usuario"	javi
'$usuario'	$usuario
$usuario	$usuario

" '$usuario' "	'javi'

Fuente: elaboración propia.

Ejecución

Existen dos formas de ejecutar los scripts:

- Anteponiendo *sh*, *source* o bien "." al nombre del script.

  root@ubuntu:~# sh ejemplo.sh

  esto es un ejemplo

  root@ubuntu:~# source ejemplo.sh

  esto es un ejemplo

  root@ubuntu:~# . ejemplo.sh

  esto es un ejemplo

- Dando permiso de ejecución y a continuación, invocándolo con su nombre anteponiendo la ruta donde se encuentra el script-:

  root@ubuntu:~# chmod +x ejemplo.sh

  root@ubuntu:~# ./ejemplo.sh

  Si nos encontramos en el mismo directorio que el script-:

  root@ubuntu:~# . ejemplo.sh

  esto es un ejemplo

  Si no pondremos la ruta, por ejemplo:

root@ubuntu:~# /root/ejemplos/ejemplo.sh

esto es un ejemplo

En caso de que el directorio donde se encuentra el script este en el PATH, se podría ejecutar introduciendo simplemente el nombre.

Parámetros

Un *shell-script* soporta argumentos o parámetros de entrada. Desde el interior del script se referencian mediante las variables especiales $i con i=1, 2, ..., Numero Argumentos. Es decir, si se lanza el siguiente script-:

root@ubuntu:~# ejemploScript uno dos gato

Se tiene que el primer parámetro es uno, el segundo dos y el tercero gato, referenciándolos en el script como $1, $2 y $3 respectivamente. Como ya se vio anteriormente que la variable $0 hace referencia al nombre del *script*.

También se puede escribir en la *shell* con el comando "set":

root@ubuntu:~# set parametro1 parametro2 parametro3

root@ubuntu:~# echo $2

parametro2

Con el comando "*shift*", se desplaza, es decir borrar varias posiciones:

root@ubuntu:~# set parametro1 parametro2 parametro3

root@ubuntu:~# shift 2

root@ubuntu:~# echo $1

parametro3

Con "*shift* 2" hace que desaparezca el valor de $1 y $2, y que ahora $1 valga lo que estaba en $3.

Estructuras de control

En cualquier lenguaje de programación, se necesita una serie de estructuras de control que permitan modificar el flujo de ejecución de las instrucciones de nuestro script.

If

La sintaxis más sencilla será:

```
if condición
then        [bloque de comandos]
[else]
        [bloque de comandos]
fi
```

Si queremos añadir más condiciones:

```
if condicion1
then        [bloque de comandos si la condición1 se cumple]
elif condicion2
then
```

```
        [bloque de comandos si la condición2 se cumple]
    else
        [bloque de comandos si no se cumplen ni la condición1 ni la condición2 ]
    fi
```

Normalmente la condición será el resultado de evaluar una primitiva o de ejecutar un comando. Veamos un ejemplo:

```
#!/bin/bash
pin="1234"
echo "Introduzca su pin"
read -s clave
if test "$clave" = "$pin"
then
echo "Pin correcto"
echo "Acceso permitido"
else
echo "Pin incorrecto"
fi
```

En este sencillo ejemplo se nos pide que introduzcamos el pin (en nuestro caso lo hemos fijado a 1234), y dependiendo si lo introducimos bien o no, nos muestra una mensaje u otro.