

REPOSITORIO ACADÉMICO DIGITAL INSTITUCIONAL

Migración aplicación de RH (SQL Server a Oracle)

Autor: Rafael Mendoza Chagolla

**Monografía presentada para obtener el título de:
Ing. En sistemas computacionales**

**Nombre del asesor:
Aldo Israel Sandoval Monroy**

Este documento está disponible para su consulta en el Repositorio Académico Digital Institucional de la Universidad Vasco de Quiroga, cuyo objetivo es integrar, organizar, almacenar, preservar y difundir en formato digital la producción intelectual resultante de la actividad académica, científica e investigadora de los diferentes campus de la universidad, para beneficio de la comunidad universitaria.

Esta iniciativa está a cargo del Centro de Información y Documentación "Dr. Silvio Zavala" que lleva adelante las tareas de gestión y coordinación para la concreción de los objetivos planteados.

Esta Tesis se publica bajo licencia Creative Commons de tipo "Reconocimiento-NoComercial-SinObraDerivada", se permite su consulta siempre y cuando se mantenga el reconocimiento de sus autores, no se haga uso comercial de las obras derivadas.





“Migración aplicación de RH (SQL Server a Oracle)”

MONOGRAFÍA

QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN SISTEMAS COMPUTACIONALES

PRESENTA

Rafael Mendoza Chagolla

ASESOR

ALDO ISRAEL SANDOVAL MONROY

CLAVE: 16PSU0049F

ACUERDO: LIC100846

MORELIA, MICHOACÁN

AGO-2013

Dedicatoria

A mi esposa y mis hijos por su paciencia.

ÍNDICE GENERAL

OBJETIVO GENERAL	VIII
OBJETIVOS ESPECÍFICOS	VIII
LIMITACIONES	IX
DELIMITACIONES	IX
1.1 INTRODUCCIÓN	1
1.2 INTRODUCCIÓN SQL SERVER.....	1
1.4 ¿QUÉ ES SQL?	2
1.5 CARACTERÍSTICAS DEL LENGUAJE	3
1.6 CÓMO INTERPRETAR UN DIAGRAMA SINTÁCTICO.....	3
1.8 INTRODUCCIÓN VISUAL BASIC 6.0.....	5
1.10 INTRODUCCIÓN A ORACLE	8
2.1 INTRODUCCIÓN	12
2.2 PROCESO DE DESARROLLO DE BASE DE DATOS	12
2.3 REQUERIMIENTOS DE INFORMACIÓN DEL NEGOCIO	13
2.4 INTRODUCCIÓN AL MODELO DE DATOS CONCEPTUAL	13
2.5 INTRODUCCIÓN AL DISEÑO DE BASE DE DATOS	14
2.6 INTRODUCCIÓN A LA CONSTRUCCIÓN DE LA BASE DE DATOS	14
2.7 MODELO CONCEPTUAL DE DATOS.....	14
2.8 ENTIDADES.....	15
2.9 RELACIONES	16
2.9.1 RELACIÓN MUCHOS A UNO (M : 1)	16
2.9.2 RELACIÓN MUCHOS A MUCHOS (M : M)	16
2.9.3 RELACIÓN UNO A UNO (1 : 1)	16
2.10 MATRIZ DE RELACIONES	17
2.11 ANALIZAR Y MODELAR RELACIONES	18
2.11.1 DETERMINAR SI EXISTE UNA RELACIÓN.....	19
2.11.2 NOMBRAR CADA RELACIÓN.....	19
2.11.3 DETERMINAR LA OPCIONALIDAD DE UNA RELACIÓN	20
2.11.4 DETERMINAR EL GRADO DE RELACIÓN.....	20
2.11.5 VALIDAR LA RELACIÓN	20
2.12 REPRESENTACIÓN DEL DIAGRAMA ENTIDAD-RELACIÓN	20
2.13 ATRIBUTOS.....	21
2.13.1 PUNTOS IMPORTANTES A TOMAR EN CUENTA SOBRE LOS ATRIBUTOS	21
2.14 IDENTIFICADORES ÚNICOS.....	22
2.15 NORMALIZAR EL MODELO DE DATOS	22
2.15.1 REGLAS DE NORMALIZACIÓN	22
2.15.1.1 Regla de la primera forma normal	23
2.15.1.2 Regla de la segunda forma normal.....	23
2.15.1.3 Regla de la tercera forma normal	23
2.16 INTRODUCCIÓN A BASES DE DATOS RELACIONALES.....	23
2.17 LLAVES PRIMARIAS.....	24
2.18 LLAVES FORÁNEAS.....	25

2.19 INTEGRIDAD DE DATOS	25
2.19.1 CONSTRAINTS DE INTEGRIDAD DE DATOS	25
2.19.1.1 Integridad de entidades.....	26
2.19.1.2 Integridad referencial.....	26
2.19.1.3 integridad de columnas	26
2.19.1.4 Integridad definida por el usuario	26
2.20 DISEÑO DE LA BASE DE DATOS	27
2.20.1 LIBERACIÓN DEL DISEÑO DE LA BASE DE DATOS	27
2.20.2 MAPEAR ENTIDADES.....	27
2.20.3 MAPEAR ATRIBUTOS A COLUMNAS.....	27
2.20.4 MAPEAR UIDS (IDENTIFICADOR ÚNICO) A LLAVES PRIMARIAS	28
2.20.5 MAPEAR RELACIONES PARA LLAVES FORÁNEAS	28
2.21 CONCLUSIONES	28
3.1 INTRODUCCIÓN	29
3.2 INSTALACIÓN DEL SQL SERVER 2005	29
3.2.1 PREPARAR EL EQUIPO PARA INSTALAR SQL SERVER 2005	30
3.2.2 INSTALAR SQL SERVER 2005.....	30
3.2.3 CONFIGURACIÓN DE LA INSTALACIÓN DEL SQL SERVER 2005.....	30
3.3 INSTALACIÓN DE ORACLE DEVELOPER	30
3.4 CAPACITACIÓN	34
3.4.1 CAPACITACIÓN DISEÑO RELACIONAL DE BASE DE DATOS	35
3.4.2 CAPACITACIÓN SQL SERVER 2005	36
3.4.3 CAPACITACIÓN VISUAL BASIC 6.0	38
3.4.4 CAPACITACIÓN DE ORACLE.....	40
3.5 CONFIGURACIÓN DEL ORIGEN DE DATOS (ODBC)	43
3.6 CONFIGURACIÓN DEL SQL DEVELOPER	45
3.7 ASIGNACIÓN DE MÓDULOS A CADA PROGRAMADOR	47
3.7.1 PROYECTOS, REPORTES Y STORED PROCEDURE DE ADMINISTRACIÓN DE RECURSOS HUMANOS, PLANEACIÓN, ADMINISTRACIÓN DEL PAGO Y HERRAMIENTAS	48
3.8 METODOLOGÍA DE TRABAJO	48
3.9 MIGRACIÓN DEL MÓDULO DE ADMINISTRACIÓN DEL PAGO.....	50
3.10 MIGRACIÓN DEL MÓDULO DE HERRAMIENTAS	75
3.11 MIGRACIÓN DEL MÓDULO DE ADMINISTRACIÓN DE RECURSOS HUMANOS	81
3.12 MIGRACIÓN DEL MÓDULO DE PLANEACIÓN	102
3.13 PRUEBAS	110
CONCLUSIONES.....	111
ÍNDICE DE FIGURAS	113

RESUMEN

En este trabajo se hablara de la migración de una aplicación de recursos humanos llamada Lobo-RH la cual esta instalada en una empresa llamada NAFIN dicha empresa solicito una migración a la plataforma Oracle como manejador de Base de Datos, se empezara dando una introducción al lenguaje SQL, posteriormente se vera, una leve introducción a Visual Basic y por ultimo un breve repaso de Oracle.

En el capitulo dos se habla de las Bases de Datos desde como son, de que forma están constituidas, en resumen, diseño, contrucción de las mismas y las reglas que debemos seguir para generar una de manera adecuada.

En el capitulo tres se habla de la instalación de los clientes de las dos plataformas es decir SQL Server 2005 y Oracle 9i, las herramientas que se utilizaron en la migración y por ultimo ejemplos comparativos de códigos.

Para terminar se habla de trabajo a futuro y la ganancia que obtuvo la empresa para poder ofrecer una herramienta que ahora puede estar montada en dos plataformas de bases de datos como son SQL Server y Oracle con esto dar más competencia en el mercado teniendo esta solución alternativa.

PLANTEAMIENTO DEL PROBLEMA

Comentario [AISM1]: Tipo de letra la misma que todo el documento

La aplicación Lobo-RH originalmente fue desarrollada en Visual Basic 6 con el motor de base de datos SQL Server, de esa forma fue instalado en la empresa NAFIN. Debido a un cambio de políticas internas y a que dicha empresa no cuenta con el personal capacitado para la administración del motor de base de datos con el que funciona actualmente la aplicación se solicitó que se realizara una migración de plataforma teniendo a Oracle como su nuevo motor de base de datos ya que cuentan con personal totalmente capacitado para el manejo de la misma.

El primer punto es que la empresa que desarrollo el software de recursos humanos no cuenta con personal capacitado en Oracle, así que se tendrá que ir aprendiendo con forme pase el tiempo y al final se terminaran afinando todos los detalles que se vayan presentando.

El segundo punto es que solo se cuenta con 6 meses para llevar a cabo dicha migración por lo cual se deberán redoblar esfuerzos para lograr dicho objetivo.

El tercer punto es que hay un contrato en el cual la empresa se encuentra obligada a tener dicha migración en el tiempo mencionado en el punto anterior de lo contrario se hara acreedor de una falta administrativa.

El cuarto punto es que la empresa no cuenta con presupuesto para poder capacitar a su personal de manera adecuada en dicho motor de base de datos asi que tendrán que ser totalmente autodidactos en su aprendizaje.

ANTECEDENTES

Hoy en día todas las empresas, instituciones, etc. generan día a día enormes cantidades de información que resulta de suma importancia para ellos, siendo lobo software una empresa que desarrolla un sistema de recursos humanos por obvias razones sus clientes no son la excepción ya que como su nombre lo indica este sistema lleva a cabo todas las tareas que en esta área se desempeñan, hay que resaltar que el área de recursos humanos de cualquier empresa o institución cuenta y maneja información laboral de todos sus empleados, además de hacer referencia al manejo, administración, gestión o dirección del personal de la empresa.

Es por ello que realizar la migración de base de datos tiene como objetivo tener una mayor seguridad de los datos de la empresa solicitante además de un mejor manejo de información en el sistema de recursos humanos, algo también importante es poder demostrar que el sistema puede ser multifuncional en cuanto a bases de datos se refiere, ya que funciona con ambos manejadores de base de datos.

Por otra parte para poder ofrecer una mejor solución de software y poder competir dentro del mercado actual se convierte en una necesidad él migrar de Microsoft hacia otra plataforma, es decir debe de existir flexibilidad, de ser así esto sería mucho más sencillo con Oracle debido a que es multiplataforma, de lo contrario siempre estaríamos casados con Microsoft debido a que con SQL Server siempre se tiene que trabajar bajo la plataforma Microsoft.

OBJETIVOS**Objetivo General**

Mantener una mejora continua con el sistema Lobo_RH, haciéndolo una herramienta que compita en el mercado, ofreciendo una alternativa más a los clientes que cuentan con manejadores de bases de datos con Oracle.

Objetivos Específicos

- Migrar la versión de la aplicación Lobo-RH de SQL Server a Oracle.
- Cumplir con los tiempos esperados para la entrega de dicho proyecto.
- Ofrecer una solución multiplataforma.

ALCANCES Y LIMITACIONES

Limitaciones

Debido a que la empresa de desarrollo cuenta con más empresas clientes, el tiempo de la migración puede ser muy prolongado, ya que si surgen reportes de cualquiera de las otras empresas se tiene que asignar personal de la migración para atender dichos pendientes.

Delimitaciones

La migración abarcará todos los módulos que comprende el sistema de recursos humanos como son:

- Planeación
- Administración de Recurso Humano
- Administración del pago
- Herramientas

JUSTIFICACIÓN

La información es poder y como las bases de datos contienen mucha información es importante tenerla bien resguardada para que no se intenten y mucho menos se lleven a cabo accesos no autorizados. Debido a esto se deben buscar las soluciones más óptimas y poder ofrecer una solución la cual pueda ayudar a las personas a sentirse seguros.

Con la migración se darán muchos beneficios, pero el más importante es mantener la seguridad e integridad de los datos, ya que como su nombre lo dice el sistema de recursos humanos se encarga de efectuar todos los procesos que en esta área se desempeñan, además de facilitar la manera de trabajar a los clientes, hacer que el sistema se adapte al usuario final y no el usuario se adapte al sistema, que el personal encargado de monitorear y/o controlar los procesos en sus servidores se sientan más familiarizado con este manejador de base de datos y así puedan desempeñar su trabajo más fácilmente.

Poder competir con las empresas de vanguardia ofreciendo un software el cual cumpla con todas las condiciones necesarias y con las expectativas del usuario final.

CAPITULO 1 GENERALIDADES DE HERRAMIENTAS

1.1 Introducción

En este capítulo se hablara de las generalidades de las herramientas que se utilizaron durante el proceso de migración de la versión como son SQL Server, Visual Basic 6 y Oracle.

1.2 Introducción SQL Server

SQL SERVER es un conjunto de objetos eficientemente almacenados. Los objetos donde se almacena la información se denominan tablas, y éstas a su vez están compuestas de filas y columnas. En el centro está el motor, el cual procesa los comandos de la base de datos. Los procesos se ejecutan dentro del sistema operativo y entienden únicamente de conexiones y de sentencias SQL.¹

SQL SERVER incluye herramientas para la administración de los recursos que la computadora nos proporciona y los gestiona para un mejor rendimiento de la base de datos.

Los nombres de las tablas que se usarán para los ejemplos de este capítulo son las siguientes:

Empleados, oficinas, productos, pedidos, clientes y ventas.

Base de datos relacional

En una base de datos relacional, los datos se organizan en tablas. Una tabla tiene cero o más filas, cada fila contiene información de un determinado 'sujeto' de la tabla, por ejemplo en una tabla de alumnos, en una fila se tienen los datos de un alumno. Las filas en un principio están desordenadas.

Cada columna representa un campo de la tabla, sirve para almacenar una determinada información, por ejemplo en una tabla de alumnos se tendrá una columna para almacenar el nombre de los alumnos. Todos los valores de una columna determinada tiene el mismo tipo de dato, y éstos están extraídos de un conjunto de valores legales llamado dominio de la columna. A parte de los valores del dominio, en una columna puede contener el valor nulo (*NULL*) que indica que no contiene ningún valor.

¹ SQL: Structured Query Language (Lenguaje de consulta estructurado).

En una tabla no puede haber dos columnas con el mismo nombre pero ese nombre si se puede utilizar en otra tabla. Normalmente todas las tablas deben tener una clave principal definida. Una clave principal es una columna (o combinación de columnas) que permiten identificar de forma inequívoca cada fila de la tabla, por lo que no pueden haber en una tabla dos filas con el mismo valor en la columna definida como clave principal.

Una clave foránea es una columna (o combinación de columnas) que contiene un valor que hace referencia a una fila de otra tabla (en algunos casos puede ser la misma tabla). Por ejemplo, tenemos dos tablas, la de alumnos y la de cursos, en la tabla de alumnos pondríamos una columna curso; para saber en qué curso está matriculado el alumno, la columna curso en la tabla de alumnos es la clave foránea, mientras que la columna código de la tabla de cursos será la clave primaria. Una tabla tiene una única clave primaria. Una tabla puede contener cero o más claves foráneas. Cuando se define una columna como clave principal, ninguna fila de la tabla puede contener un valor nulo en esa columna tampoco se pueden repetir los valores en la columna. Cuando se define una columna como clave foránea, las filas de la tabla pueden contener en esa columna o bien el valor nulo, o bien un valor que existe en la otra tabla. Eso es lo que se denomina integridad referencial que consiste en que los datos que referencian otros (clave foránea) deben de ser correctos.

1.4 ¿Qué es SQL?

El SQL (*Structured Query Language*), Lenguaje de Consulta Estructurado, es un lenguaje surgido de un proyecto de investigación de IBM (*International Business Machines*) para el acceso a bases de datos relacionales. Actualmente se ha convertido en un estándar de lenguaje de bases de datos, y la mayoría de los sistemas de bases de datos lo soportan desde sistemas para computadoras personales, hasta grandes computadoras. Por supuesto, a partir del estándar cada sistema ha desarrollado su propio SQL que puede variar de un sistema a otro, pero con cambios que no suponen ninguna complicación para alguien que conozca un SQL concreto.

Como su nombre indica, el SQL nos permite realizar consultas a la base de datos. Pero el nombre queda corto ya que SQL además realiza funciones de definición, control y gestión

de la base de datos. Las sentencias SQL se clasifican según su finalidad dando origen a tres sub lenguajes:

- El **DDL** (*Data Description Language*), lenguaje de definición de datos, incluye órdenes para definir, modificar o borrar las tablas en las que se almacenan los datos y de las relaciones entre estas (es el que más varía de un sistema a otro).
- El **DCL** (*Data Control Language*), lenguaje de control de datos, contiene elementos útiles para trabajar en un entorno multiusuario, en el que es importante la protección de los datos, la seguridad de las tablas y establecimiento de restricciones en el acceso, así como elementos para coordinar la compartición de datos por parte de usuarios concurrentes, asegurando que no interfieran unos con otros.
- El **DML** (*Data Manipulation Language*), lenguaje de manipulación de datos, nos permite recuperar los datos almacenados de la base de datos y también incluye órdenes para permitir al usuario actualizar la base de datos añadiendo nuevos datos, suprimiendo datos antiguos o modificando datos previamente almacenados.

1.5 Características del lenguaje

Una sentencia SQL es como una frase (escrita en inglés) con la que decimos lo que se quiere obtener y de donde obtenerlo. Todas las sentencias empiezan con un verbo (palabra reservada que indica la acción a realizar), seguido del resto de cláusulas, algunas obligatorias y otras opcionales que completan la frase. Todas las sentencias siguen una sintaxis para que se puedan ejecutar correctamente, para describir esa sintaxis utilizaremos un diagrama sintáctico como el que se muestra a continuación.

1.6 Cómo interpretar un diagrama sintáctico

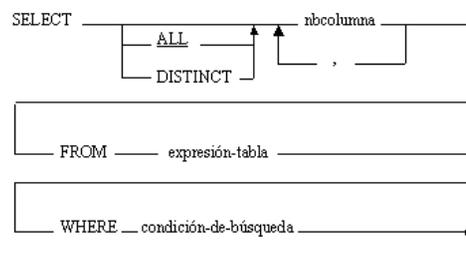


Fig. 1.0 interpretación de un diagrama sintáctico

Las palabras que aparecen en mayúsculas son las palabras reservadas, se tiene que poner tal cual y no se pueden utilizar para otro fin, por ejemplo, en el diagrama de la figura tenemos las palabras reservadas *select*, *all*, *distinct*, *from* *where*. Las palabras en minúsculas son variables que el usuario deberá sustituir por un dato concreto. En el diagrama se tiene nb columna, expresión-tabla y condicion-de-busqueda.

Una sentencia válida se construye siguiendo la línea a través del diagrama hasta el punto que marca el final. Las líneas se siguen de izquierda a derecha y de arriba abajo. Cuando se quiere alterar el orden normal se indica con una flecha.

Microsoft **SQL Server**

Data Types Exact Numerics bit decimal tinyint money smallint bit int numeric bigint int Approximate Numerics float real Date and Time smalldatetime timestamp datetime Strings char text varchar Unicode Strings nchar ntext nvarchar Binary Strings binary image varbinary Miscellaneous cursor table sql_variant xml	Date Functions DATEADD (datepart, number, date) DATEDIFF (datepart, start, end) DATENAME (datepart, date) DATEPART (datepart, date) DAY (date) GETDATE () GETUTCDATE () MONTH (date) YEAR (date)	Create a Stored Procedure CREATE PROCEDURE name @variable AS datatype = value AS -- Comments SELECT * FROM table GO
Type Conversion CAST (expression AS datatype) CONVERT (datatype, expression)	Dateparts Year yy, YYYY Quarter qq, q Month mm, m Day of Year dy, y Day dd, d Week wk, ww Hour hh Minute mi, n Second ss, s Millisecond ms	Create a Trigger CREATE TRIGGER name ON table FOR DELETE, INSERT, UPDATE AS -- Comments SELECT * FROM table GO
Ranking Functions RANK NTILE DENSE_RANK ROW_NUMBER	Mathematical Functions ABS LOG10 ACOS PI ASIN POWER ATAN RADIANS ATN2 RAND CEILING ROUND COS SIGN COT SIN DEGREES SQUARE EXP SQRT FLOOR TAN LOG	Create a View CREATE VIEW name AS -- Comments SELECT * FROM table GO
Grouping (Aggregate) Functions AVG MAX BINARY_CHECKSUM MIN CHECKSUM SUM CHECKSUM_AVG STDEV COUNT STDEVP COUNT_BIG VAR GROUPING VARP	String Functions ASCII REPLICATE CHAR REVERSE CHARINDEX RIGHT DIFFERENCE RTRIM LEFT SOUNDEX LEN SPACE LOWER STR LTRIM STUFF NCHAR SUBSTRING PATINDEX UNICODE REPLACE UPPER QUOTENAME	Create an Index CREATE UNIQUE INDEX name ON table (columns)
Table Functions ALTER DROP CREATE TRUNCATE		Create a Function CREATE FUNCTION name (@variable datatype(length)) RETURNS datatype(length) AS BEGIN DECLARE @return datatype(length) SELECT @return = CASE @variable WHEN 'a' THEN 'return a' WHEN 'b' THEN 'return b' ELSE 'return c' RETURN @return END

Fig. 1.1 Sheet of cheats SQL Server

1.8 Introducción visual Basic 6.0

Visual Basic es un ambiente grafico de desarrollo de aplicaciones para el sistema operativo Microsoft Windows. Las aplicaciones están basadas en objetos y son manejadas por eventos. Este lenguaje se deriva del lenguaje Basic, el cual es un lenguaje de programación estructurado. Sin embargo, el lenguaje de programación emplea un modelo de programación manejada por eventos.

Visual Basic Quick Reference

Operators

+, -, *, /	Addition, subtraction, multiplication, division
\	Integer Division
Mod	Remainder
^	Exponent
&	String concatenation
=, >, <, >=, <=	Comparison
NOT, AND, OR	Boolean operators

Data Types

Variant, Integer (%), Long (L), Single (S), Double (D), Byte, Boolean, Date, Currency (C), String (S)

CBool(expr), CByte(expr), CChar(expr), CDate(expr), CDbl(expr), CDec(expr), CInt(expr), CLng(expr), CSng(expr), CStr(expr), CVar(expr)

[Public | Private] Const *constname* [As *type*] = *expression*

Dim [WithEvents] *varname* [[*subscript*]] [As *[New]* *type*]

ReDim [Preserve] *varname* [[*subscript*]] [As *type*]

[Public | Private] Enum *name*

[Private | Public] Type *varname*
elementname [[*subscript*]] As *type*
[[*elementname*]] [[*subscript*]] As *type*
...
End Type

Set *objectvar* = [[*New*] *objectexpression* | Nothing]

Static *varname* [[*subscript*]] [As *[New]* *type*]

Math Functions

Abs(*num*), Atn(*num*), Cos(*num*), Log(*num*), Rnd(*num*), Randomize, Sin(*num*), Sqr(*num*), Tan(*num*)

FV(*rate*, *nper*, *pmt*, *pv*, [*type*])

NPV(*rate*, *values*)

PV(*rate*, *nper*, *pmt*, *f*, [*type*])

PMT(*rate*, *nper*, *pv*, *f*, [*type*])

PPMT(*rate*, *per*, *nper*, *pv*, *f*, [*type*])

String Functions

Left(*string*, *length*), Right(*string*, *length*), Mid(*string*, *start*, *length*)

UCase(*string*), LCase(*string*), Len(*string*)

LTrim(*string*), RTrim(*string*), Trim(*string*)

Asc(*string*), Val(*string*), Oct(*number*), Hex(*number*)

Split(*expression*, [*delimiter*], [*count*], [*compare*])
Join(*list*, [*delimiter*])

Replace(*expression*, *find*, *replacewith*, [*start*], [*count*], [*compare*])

StrComp(*string1*, *string2*, [*compare*])

Filter(*inputstrings*, *values*, [*include*], [*compare*])

StrReverse(*string1*)

InStr(*start*, *string1*, *string2*, [*compare*])

InStrRev(*string1*, *string2*, *start*, [*compare*])

Program Flow

For *counter* = *start* To *end* [Step *step*]

[*statements*]

[Exit For]

[*statements*]

Next [*counter*]

For Each *element* In *group*

[*statements*]

[Exit For]

[*statements*]

Next [*element*]

If *condition* Then [*statements*] [Else *elsestatements*]

Or, you can use the block form syntax:

If *condition* Then

[*statements*]

[Elseif *condition-n* Then

[*statements*]

[Else

[*statements*]

End If

Do [While | Until]

condition

[*statements*]

[Exit Do]

[*statements*]

Loop

Select Case *stateexpression*

[Case *expressionlist-n*

[*statements*]] . . .

[Case Else

[*statements*]]

End Select

While *condition*

[*statements*]

Wend

With *object*

[*statements*]

End With

On Error GoTo *line*

On Error Resume [0|1|2|Next]

On Error GoTo 0

File Operation

Open *pathname* For mode [Access access] [lock] As #*n*

filenumber [Len=*reclength*]

Input #*filenumber*, *varlist*

Print #*filenumber*, [*outputlist*]

Line Input #*filenumber*, *varname*

Write #*filenumber*, [*outputlist*]

Get #*filenumber*, [*recnumber*], *varname*

Put #*filenumber*, [*recnumber*], *varname*

Loc #*filenumber*

Seek #*filenumber*, *position*

Eof #*filenumber*

LoF #*filenumber*

Reset

Close #*filenumberlist*

Lock #*filenumber*, [*recordrange*]

Unlock #*filenumber*, [*recordrange*]

Function and Procedure

[Public | Private | Friend] [Static] Function *name* [(*arglist*)] [As *type*]

[*statements*]

[Exit Function]

[*statements*]

End Function

[Private | Public | Friend] [Static] Sub *name* [(*arglist*)]

[*statements*]

[Exit Sub]

[*statements*]

End Sub

[Public | Private] Declare Sub *name* Lib "*libname*" [Alias "*aliasname*"] [(*arglist*)]

[Public | Private] Declare Function *name* Lib "*libname*" [Alias "*aliasname*"] [(*arglist*)] [As *type*]

[Call] *name* [(*ByVal*) *argumentlist*]

Property Procedure

[Public | Private | Friend] [Static] Property Get *name*

[(*arglist*)] [As *type*]

[*statements*]

[Exit Property]

[*statements*]

End Property

Fig. 1.2 Sheet of cheats Visual Basic 6

<pre>[Public Private Friend] [Static] Property Let name ([arglist], value) [statesash] [Exit Property] [statesash] End Property [Public Private Friend] [Static] Property Set name ([arglist], reference) [statesash] [Exit Property] [statesash] End Property</pre>	<pre>Properties: Comments, CompanyName, EXENAME, FileDescription, HelpFile, LegalCopyright, LegalTrademarks, LogMode, LogPath, Major, Minor, NonModalAllowed, OLERequestPendingMsgText, OLERequestPendingMsgTitle, OLERequestPendingTimeout, OLEServerBusyMsgText, OLEServerBusyMsgTitle, OLEServerBusyRaiseError, OLEServerBusyTimeout, Path, PreInstance, ProductName, RetainedProject, Revision, StartMode, TaskVisible, ThreadID, Title, UnattendedApp, hInstance Methods: LogEvent, StartLogging</pre>	<pre>KeyUp, KeyPress, LostFocus, MouseDown, MouseDown, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag, Validate</pre>
<p>System and Miscellaneous</p> <pre>AppActivate int[, wait] Make a windows focus SendKeys string[, wait] Send key strokes to current app Shell(pathname[, windowstyle]) Run an external program seconds elapsed since midnight Timer Command line Beep Beep on internal speaker Army base Option Base (0 1) Force explicit declaration CNDir path Change current directory MKDir path Make a directory RndM path Random a directory ChDrive drive Change current drive Kill filename Delete a file FileCopy source, destination Copy a file Name oldAs new Rename a file FileLen(pathname) File size FileDateTime(pathname) File creation data Date Get/Set system date Time Get/Set system time Environ([environment number]) Environment string Error([error number]) Error description string * comment space Constant line InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context]) MsgBox(prompt[, buttons] [, title] [, helpfile, context])</pre>	<pre>Printer Properties: ColorMode, Copies, Count, CurrentX, CurrentY, DeviceName, DrawMode, DrawStyle, DrawWidth, DriverName, Duplex, FillColor, FillStyle, Font, FontBold, FontItalic, FontStrikeThru, FontUnderline, FontCount, FontName, FontSize, FontTransparent, Fonts, Height, Width, Orientation, Page, PaperWdth, PaperSize, Port, PrintQuality, RightToLeft, ScaleHeight, ScaleWidth, ScaleLeft, ScaleTop, ScaleMode, TrackDefault, TwipsPerPointX, TwipsPerPointY, Zoom, HDC Methods: Circle, EndDoc, KillDoc, Line, NewPage, PGSet, PaintPicture, Scale, ScaleX, ScaleY, TextHeight, TextWidth Timer Properties: Enabled, Index, Interval, Left, Top, Name, Parent, Tag</pre>	<p>ComboBox</p> <pre>Properties: Appearance, BackColor, ForeColor, Container, DataChanged, DataField, DataFormat, DataMember, DragIcon, DragMode, Enabled, Font, FontBold, FontItalic, FontStrikeThru, FontUnderline, FontName, FontSize, Height, Width, HelpContextID, Index, IntegralHeight, ItemData, Left, Top, List, ListCount, ListIndex, Locked, MouseIcon, MousePointer, Name, NewIndex, OLEDragMode, OLEDropMode, Parent, RightToLeft, RightToLeft, SelLength, SelStart, SelText, SelLength, SelStart, SelText, Sorted, Style, TabIndex, TabStop, Tag, Text, ToolTipText, TopIndex, Visible, WhatsThisHelpID, HWND Methods: AddItem, Clear, Drag, Move, OLEDrag, Refresh, RemoveItem, SetFocus, ShowWhatsThis, ZOrder Events: Change, Click, DblClick, DragDrop, DragOver, DropDown, GotFocus, KeyDown, KeyUp, KeyPress, LostFocus, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag, Scroll, Validate</pre>
<pre>GetSetting(appname, section, key[, default]) SaveSetting(appname, section, key, setting) GetAllSettings(appname, section) DeleteSetting(appname, section, key) CreatesObject(class, [servername]) GetObject([pathname] [, class]) Load Object Unload object LoadPicture(filename[, bits], [colordepth], [v]) SavePicture picture, stringexpression</pre>	<p>Standard Controls</p> <p>CheckBox</p> <pre>Properties: Alignment, Appearance, BackColor, ForeColor, Caption, Container, DataChanged, DataField, DataFormat, DataMember, DisabledPicture, DownPicture, DragIcon, DragMode, Enabled, Font, FontBold, FontItalic, FontStrikeThru, FontUnderline, FontName, FontSize, Height, Width, HelpContextID, Index, Left, Top, MaskColor, MouseIcon, MousePointer, Name, OLEDropMode, Parent, Picture, RightToLeft, OLEDropMode, Parent, Picture, RightToLeft, RightToLeft, Style, TabIndex, TabStop, Tag, ToolTipText, UseMaskColor, Value, Visible, WhatsThisHelpID, HWND Methods: Drag, Move, OLEDrag, Refresh, SelfFocus, ShowWhatsThis, ZOrder Events: Click, DragDrop, DragOver, GotFocus, KeyDown,</pre>	<p>CommandButton</p> <pre>Properties: Appearance, BackColor, ForeColor, Cancel, Caption, Container, Default, DisabledPicture, DownPicture, DragIcon, DragMode, Enabled, Font, FontBold, FontItalic, FontStrikeThru, FontUnderline, FontName, FontSize, Height, Width, HelpContextID, Index, Left, Top, MaskColor, MouseIcon, MousePointer, Name, OLEDropMode, Parent, Picture, RightToLeft, UseMaskColor, Value, Visible, WhatsThisHelpID, HWND Methods: Drag, Move, OLEDrag, Refresh, ShowWhatsThis, UpdateControls, UpdateRecords ZOrder Events: Click, DragDrop, DragOver, GotFocus, KeyDown, KeyUp, KeyPress, LostFocus, MouseDown, MouseDown, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag</pre>
<p>Objects</p> <pre>App</pre>	<p>Data</p> <pre>Properties: Align, Appearance, BOFAction, EOFAction, BackColor, ForeColor, Caption, Connect, Database, DatabaseName, DefaultCursorType,</pre>	

Fig. 1.2 Sheet of cheats Visual Basic 6

<p>DefaultType, DragIcon, DragMode, EditMode, Enabled, Exclusive, Font, FontBold, FontItalic, FontStrikethru, FontUnderline, FontName, FontSize, Height, Width, Index, Left, Top, MouseIcon, MousePointer, Name, OLEDropMode, Options, Parent, ReadOnly, RecordSource, Recordset, RecordsetType, RightToLeft, RightToLeft, Tag, ToolTipText, Visible, WhatsThisHelpID</p> <p>Methods: Drag, Move, OLEDrag, Refresh, ShowWhatsThis, UpdateControls, UpdateRecord, ZOrder</p> <p>Events: DragDrop, DragOver, Error, MouseDown, MouseUp, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag, Reposition, Resize, Validate</p>	<p>FileListBox</p> <p>Properties: Appearance, Archive, Hidden, Normal, System, BackColor, ForeColor, Container, DragIcon, DragMode, Enabled, FileName, Font, FontBold, FontItalic, FontStrikethru, FontUnderline, FontName, FontSize, Height, Width, HelpContextID, Index, Left, Top, List, ListCount, ListIndex, Looked, MouseIcon, MousePointer, MultiSelect, Name, OLEDragMode, OLEDropMode, Parent, Path, Pattern, ReadOnly, Selected, TabIndex, TabStop, Tag, ToolTipText, TopIndex, Visible, WhatsThisHelpID, hWnd</p> <p>Methods: Drag, Move, OLEDrag, Refresh, SelfFocus, ShowWhatsThis, ZOrder</p> <p>Events: Click, DblClick, DragDrop, DragOver, GotFocus, KeyDown, KeyUp, KeyPress, LostFocus, MouseDown, MouseUp, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag, PathChange, PatternChange, Scroll, Validate</p>	<p>KeyDown, KeyUp, KeyPress, LostFocus, Scroll, Validate</p>
<p>DirListBox</p> <p>Properties: Appearance, BackColor, ForeColor, Container, DragIcon, DragMode, Enabled, Font, FontBold, FontItalic, FontStrikethru, FontUnderline, FontName, FontSize, Height, Width, HelpContextID, Index, Left, Top, List, ListCount, ListIndex, MouseIcon, MousePointer, Name, OLEDropMode, OLEDropMode, Parent, Path, TabIndex, TabStop, Tag, ToolTipText, TopIndex, Visible, WhatsThisHelpID, hWnd</p> <p>Methods: Drag, Move, OLEDrag, Refresh, SelfFocus, ShowWhatsThis, ZOrder</p> <p>Events: Change, Click, DragDrop, DragOver, GotFocus, KeyDown, KeyUp, KeyPress, LostFocus, MouseDown, MouseUp, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag, Scroll, Validate</p>	<p>Frame</p> <p>Properties: Appearance, BackColor, ForeColor, BorderStyle, Caption, ClipControls, Container, DragIcon, DragMode, Enabled, Font, FontBold, FontItalic, FontStrikethru, FontUnderline, FontName, FontSize, Height, Width, HelpContextID, Index, Left, Top, MouseIcon, MousePointer, Name, OLEDropMode, Parent, RightToLeft, TabIndex, Tag, ToolTipText, Visible, WhatsThisHelpID, hWnd</p> <p>Methods: Drag, Move, OLEDrag, Refresh, ShowWhatsThis, ZOrder</p> <p>Events: Click, DblClick, DragDrop, DragOver, MouseDown, MouseUp, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag</p>	<p>Image</p> <p>Properties: Appearance, BorderStyle, Container, DataChanged, DataField, DataFormat, DataMember, DataSource, DragIcon, DragMode, Enabled, Height, Width, Index, Left, Top, MouseIcon, MousePointer, Name, OLEDragMode, OLEDropMode, Parent, Picture, Stretch, Tag, ToolTipText, Visible, WhatsThisHelpID</p> <p>Methods: Drag, Move, OLEDrag, Refresh, ShowWhatsThis, ZOrder</p> <p>Events: Click, DblClick, DragDrop, DragOver, MouseDown, MouseUp, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag</p>
<p>DriveListBox</p> <p>Properties: Appearance, BackColor, ForeColor, Container, DragIcon, DragMode, Drive, Enabled, Font, FontBold, FontItalic, FontStrikethru, FontUnderline, FontName, FontSize, Height, Width, HelpContextID, Index, Left, Top, List, ListCount, ListIndex, MouseIcon, MousePointer, Name, OLEDropMode, Parent, TabIndex, TabStop, Tag, ToolTipText, TopIndex, Visible, WhatsThisHelpID, hWnd</p> <p>Methods: Drag, Move, OLEDrag, Refresh, SelfFocus, ShowWhatsThis, ZOrder</p> <p>Events: Change, DragDrop, DragOver, GotFocus, KeyDown, KeyUp, KeyPress, LostFocus, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag, Scroll, Validate</p>	<p>HScrollBar/ScrollBar</p> <p>Properties: Container, DragIcon, DragMode, Enabled, Height, Width, HelpContextID, Index, LargeChange, SmallChange, Left, Top, Max, Min, MouseIcon, MousePointer, Name, Parent, RightToLeft, TabIndex, TabStop, Tag, Value, Visible, WhatsThisHelpID, hWnd</p> <p>Methods: Drag, Move, Refresh, SelfFocus, ShowWhatsThis, ZOrder</p> <p>Events: Change, DragDrop, DragOver, GotFocus,</p>	<p>Label</p> <p>Properties: Alignment, Appearance, AutoSize, BackColor, ForeColor, BackStyle, BorderStyle, Caption, Container, DataChanged, DataField, DataFormat, DataMember, DataSource, DragIcon, DragMode, Enabled, Font, FontBold, FontItalic, FontStrikethru, FontUnderline, FontName, FontSize, Height, Width, Index, Left, Top, LinkItem, LinkMode, LinkTimeout, LinkTopic, MouseIcon, MousePointer, Name, OLEDrag Method, OLEDropMode, Parent, RightToLeft, TabIndex, Tag, ToolTipText, UseMnemonic, Visible, WhatsThisHelpID, WordWrap</p> <p>Methods: Drag, LinkExecute, LinkPoke, LinkRequest, LinkSend, Move, OLEDrag, Refresh, ShowWhatsThis, ZOrder</p> <p>Events: Change, Click, DblClick, DragDrop, DragOver, LinkClose, LinkDrop, LinkNotify, LinkOpen, MouseDown, MouseUp, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag</p>
		<p>Line</p> <p>Properties: BorderColor, BorderStyle, BorderWidth, Container, DrawMode, Index, Name, Parent, Tag, Visible, X1, Y1, X2, Y2</p> <p>Methods: Refresh, ZOrder</p>
		<p>ListBox</p> <p>Properties: Appearance, BackColor, ForeColor, Columns, Container, DataChanged, DataField, DataFormat,</p>

Fig. 1.2 Sheet of cheats Visual Basic 6

<p>DataMember, DataSource, DragIcon, DragMode, Enabled, Font, FontBold, FontItalic, FontStrikeThru, FontUnderline, FontName, FontSize, Height, Width, HelpContextID, Index, IntegralHeight, ItemData, Left, Top, List, ListCount, ListIndex, MouseIcon, MousePointer, MultiSelect, Name, NewIndex, OLEDragMode, OLEDropMode, Parent, RightToLeft, ScrollCount, Selected, Sorted, Style, TabIndex, TabStop, Tag, Text, ToolTipText, TopIndex, Visible, WhatsThisHelpID, hWnd</p> <p>Methods: AddItem, Clear, Drag, Move, OLEDrag, Refresh, RemoveItem, SelfFocus, ShowWhatsThis, ZOrder</p> <p>Events: Click, DbClick, DragDrop, DragOver, GotFocus, ItemCheck, KeyDown, KeyUp, KeyPress, LostFocus, MouseDown, MouseUp, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLEGetData, OLEStartDrag, Scroll, Validate</p> <p>Menu</p> <p>Properties: Caption, Checked, Enabled, HelpContextID, Index, Name, NegotiatePosition, Parent, Shortcut, Tag, Visible, WindowList</p> <p>Events: Click</p> <p>OptionButton</p> <p>Properties: Alignment, Appearance, BackColor, ForeColor, Caption, Container, DateFormat, DisabledPicture, DownPicture, DragIcon, DragMode, Enabled, Font, FontBold, FontItalic, FontStrikeThru, FontUnderline, FontName, FontSize, Height, Width, HelpContextID, Index, Left, Top, MaskColor, MouseIcon, MousePointer, Name, OLEDropMode, Parent, Picture, RightToLeft, Style, TabIndex, TabStop, Tag, ToolTipText, UseMaskColor, Value, Visible, WhatsThisHelpID, hWnd</p> <p>Methods: Drag, Move, OLEDrag, Refresh, SelfFocus, ShowWhatsThis, ZOrder</p> <p>Events: Click, DbClick, DragDrop, DragOver, GotFocus, KeyDown, KeyUp, KeyPress, LostFocus, MouseDown, MouseUp, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLEGetData, OLEStartDrag, Validate</p>	<p>PictureBox</p> <p>Properties: Align, Appearance, AutoRedraw, AutoSize, BackColor, ForeColor, BorderStyle, ClipControls, Container, CurrentX, CurrentY, DataChanged, DataField, DateFormat, DataMember, DataSource, DragIcon, DragMode, DrawMode, DrawStyle, DrawWidth, Enabled, FillColor, FillStyle, Font, FontBold, FontItalic, FontStrikeThru, FontUnderline, FontName, FontSize, FontTransparent, Height, Width, HelpContextID, Image, Index, Left, Top, LinkItem, LinkMode, LinkTimeout, LinkTopic, MouseIcon, MousePointer, Name, Negotiate, OLEDragMode, OLEDropMode, Parent, Picture, RightToLeft, ScaleHeight, ScaleWidth, ScaleLeft, ScaleTop, ScaleMode, TabIndex, TabStop, Tag, ToolTipText, Visible, WhatsThisHelpID, hDC, hWnd</p> <p>Methods: Circle, Cls, Drag, Line, LinkExecute, LinkPoke, LinkRequest, LinkSend, Move, OLEDrag, PSet, PaintPicture, Point, Refresh, Scale, ScaleX, ScaleY, SelfFocus, ShowWhatsThis, TextHeight, TextWidth, ZOrder</p> <p>Events: Change, Click, DbClick, DragDrop, DragOver, GotFocus, KeyDown, KeyUp, KeyPress, LinkClose, LinkError, LinkNotify, LinkOpen, LostFocus, MouseDown, MouseUp, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLEGetData, OLEStartDrag, Paint, Resize, Validate</p> <p>Shape</p> <p>Properties: BackColor, ForeColor, BackStyle, BorderColor, BorderStyle, BorderWidth, Container, DrawMode, FillColor, FillStyle, Height, Width, Index, Left, Top, Name, Parent, Shape, Tag, Visible</p> <p>Methods: Move, Refresh, Zorder</p> <p>TextBox</p> <p>Properties: Alignment, Appearance, BackColor, ForeColor, BorderStyle, Container, DataChanged, DataField, DateFormat, DataMember, DataSource, DragIcon, DragMode, Enabled, Font, FontBold, FontItalic, FontStrikeThru, FontUnderline, FontName, FontSize, Height, Width, HelpContextID, HideSelection, Index, Left, Top, LinkItem, LinkMode, LinkTimeout, LinkTopic, Locked, MaxLength, MouseIcon, MousePointer, MultiLine, Name, OLEDragMode, OLEDropMode, Parent, PasswordChar, RightToLeft, ScrollBars, SelLength, SelStart, SelText, TabIndex, TabStop, Tag, Text, ToolTipText, Visible, WhatsThisHelpID, hWnd</p>	<p>Methods: Drag, LinkExecute, LinkPoke, LinkRequest, LinkSend, Move, OLEDrag, Refresh, SelfFocus, ShowWhatsThis, Zorder</p> <p>Events: Change, Click, DbClick, DragDrop, DragOver, GotFocus, KeyDown, KeyUp, KeyPress, LinkClose, LinkError, LinkNotify, LinkOpen, LostFocus, MouseDown, MouseUp, MouseMove, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLEGetData, OLEStartDrag, Validate</p> <p>OLE Container</p> <p>Properties: Action, AppleRunning, Appearance, AutoActivate, AutoVerbMenu, BackColor, ForeColor, BackStyle, BorderStyle, Class, Container, Data, DataChanged, DataField, DataText, DisplayType, DragIcon, DragMode, Enabled, FileName, Format, Height, Width, HelpContextID, HostName, Index, Left, Top, MiscFlags, MouseIcon, MousePointer, Name, OLEDropAllowed, OLEType, OLETypeAllowed, Object, ObjectAcceptFormats, ObjectGetFormatsCount, ObjectVerbFlags, ObjectVerbs, ObjectVerbsCount, Parent, PasteOK, Picture, SizeMode, SourceDoc, SourceItem, TabIndex, TabStop, Tag, UpdateOptions, Verb, Visible, WhatsThisHelpID, hWnd, ipoleObject</p> <p>Methods: Close, Copy, CreateEmbed, CreateLink, Delete, DoVerb, Drag, FetchVerbs, InsertODIG, Move, Paste, PasteSpecial(Dlg, ReadFromFile, Refresh, SaveToFile, SaveToOLEFile, SelfFocus, ShowWhatsThis, Update, ZOrder</p> <p>Events: Click, DbClick, DragDrop, DragOver, GotFocus, KeyDown, KeyUp, KeyPress, LostFocus, MouseDown, MouseUp, MouseMove, ObjectMove, Resize, Updated</p> <p>Common Dialog</p> <p>Properties: Action, CancelError, Color, Copies, DefaultExt, DialogTitle, FileName, FileTitle, Filter, FilterIndex, Flags, FontBold, FontItalic, FontStrikeThru, FontName, FontSize, FromPage, ToPage, hDC, HelpCommand, HelpContext, HelpFile, HelpKey, Index, INDI, Left, Top, Max, Min, MaxFileSize, Name, Object, Orientation, Parent, PrinterDefault</p> <p>Methods: AboutBox, ShowColor, ShowFont, ShowHelp, ShowOpen, ShowPrinter, ShowSave</p>
--	--	---

Fig. 1.2 Sheet of cheats Visual Basic 6

1.10 Introducción a Oracle

Oracle es una herramienta cliente/servidor para la gestión de base de datos. Es un producto que por su gran potencia y elevado costo hace que solo se vea en empresas muy grandes y multinacionales. De igual manera pasa en el desarrollo Web, debido a que es un sistema demasiado caro no está tan extendido como otras bases de datos como Access, MySQL, SQL SERVER, etc.

ORACLE SQL REFERENCE CARD

Reserved Words	Built-in Functions	Common Clauses/Misc Cmds	ALTER / DROP COMMANDS	CREATE Commands
ACCESS	ABS	allocate extent clause	ALTER CLUSTER	CREATE CLUSTER
ADD	ACOS	constraints	ALTER DATABASE	CREATE CONTEXT
ALL	ADD_MONTHS	deallocate unused clause	ALTER DIMENSION	CREATE CONTROLFILE
ALTER	ASCII	file specification	ALTER FUNCTION	CREATE DATABASE
AND	ASCIISTR	logging clause	ALTER INDEX	CREATE DATABASE LINK
ANY	ASIN	parallel clause	ALTER INDEXTYPE	CREATE DIMENSION
AS	ATAN	physical_attributes_clause	ALTER JAVA	CREATE DIRECTORY
ASC	ATAN2	storage clause	ALTER MATERIALIZED VIEW	CREATE FUNCTION
AUDIT	AVG		ALTER MATERIALIZED VIEW LOG	CREATE INDEX
BETWEEN	BFILENAME		ALTER OPERATOR	CREATE INDEXTYPE
BY	BIN_TO_NUM		ALTER OUTLINE	CREATE JAVA
CHAR	BITAND	MISCELLANEOUS COMMANDS	ALTER PACKAGE	CREATE LIBRARY
CHECK	CAST	ANALYZE	ALTER PROCEDURE	CREATE MATERIALIZED VIEW
CLUSTER	CEIL	ASSOCIATE STATISTICS	ALTER PROFILE	CREATE MATERIALIZED VIEW LOG
COLUMN	CHARTOROWID	AUDIT	ALTER RESOURCE COST	CREATE OPERATOR
COMMENT	CHR	CALL	ALTER ROLE	CREATE OUTLINE
COMPRESS	COALESCE	COMMENT	ALTER ROLLBACK SEGMENT	CREATE PACKAGE
CONNECT	COMPOSE	COMMIT	ALTER SEQUENCE	CREATE PACKAGE BODY
CREATE	CONCAT	DELETE	ALTER SESSION	CREATE PFILE
CURRENT	CONVERT	DISASSOCIATE STATISTICS	ALTER SYSTEM	CREATE PROCEDURE
DATE	CORR	EXPLAIN PLAN	ALTER TABLE	CREATE PROFILE
DECIMAL	COS	GRANT	ALTER TABLESPACE	CREATE ROLE
DEFAULT	COSH	INSERT	ALTER TRIGGER	CREATE ROLLBACK SEGMENT
DELETE	COUNT	LOCK TABLE	ALTER TYPE	CREATE SCHEMA
DESC	COVAR_POP	MERGE	ALTER USER	CREATE SEQUENCE
DISTINCT	COVAR_SAMP	NOAUDIT	ALTER VIEW	CREATE SPFILE
DROP	CUME_DIST	RENAME		CREATE SYNONYM
ELSE	CURRENT_DATE	REVOKE	DROP COMMANDS	CREATE TABLE
EXCLUSIVE	CURRENT_TIMESTAMP	ROLLBACK	DROP CLUSTER	CREATE TABLESPACE
EXISTS	DBTIMEZONE	SAVEPOINT	DROP CONTEXT	CREATE TEMPORARY TABLESPACE
FILE	DECODE	SELECT	DROP DATABASE LINK	CREATE TRIGGER
FLOAT	DECOMPOSE	SET CONSTRAINTS	DROP DIMENSION	CREATE TYPE
FOR	DENSE_RANK	SET ROLE	DROP DIRECTORY	CREATE TYPE BODY
FROM	DEPTH	SET TRANSACTION		CREATE USER

Fig. 1.3 Sheet of cheats Oracle

Reserved Words	Built-in Functions	Common Clauses/Misc Cmds	ALTER / DROP COMMANDS	CREATE Commands
GRANT	DEREF	TRUNCATE	DROP FUNCTION	CREATE VIEW
GROUP	DUMP	UPDATE	DROP INDEX	
HAVING	EMPTY_BLOB_CLOB		DROP INDEXTYPE	
IDENTIFIED	EXISTSNODE		DROP JAVA	
IMMEDIATE	EXP	DML COMMANDS	DROP LIBRARY	PSUEDO COLUMNS
IN	EXTRACT (datetime)	COMMIT	DROP MATERIALIZED VIEW	CURRVAL and NEXTVAL
INCREMENT	EXTRACT (XML)	DELETE	DROP MATERIALIZED VIEW LOG	LEVEL
INDEX	EXTRACTVALUE	INSERT	DROP OPERATOR	ROWID
INITIAL	FIRST	MERGE	DROP OUTLINE	ROWNUM
INSERT	FIRST_VALUE	ROLLBACK	DROP PACKAGE	XMLDATA
INTEGER	FLOOR	SAVEPOINT	DROP PROCEDURE	
INTERSECT	FROM_TZ	SELECT	DROP PROFILE	
INTO	GREATEST	TRUNCATE	DROP ROLE	DB OBJECTS (SCHEMA)
IS	GROUP_ID	UPDATE	DROP ROLLBACK SEGMENT	Clusters
LEVEL	GROUPING		DROP SEQUENCE	Constraints
LIKE	GROUPING_ID		DROP SYNONYM	Database links
LOCK	HEXTORAW	DCL COMMANDS	DROP TABLE	Database triggers
LONG	INITCAP	AUDIT	DROP TABLESPACE	Dimensions
MAXEXTENTS	INSTR	GRANT	DROP TRIGGER	External procedure libraries
MINUS	LAG	NOAUDIT	DROP TYPE	Index-organized tables
MLSLABEL	LAST	REVOKE	DROP TYPE BODY	Indexes
MODE	LAST_DAY		DROP USER	Indextypes
MODIFY	LAST_VALUE	DDL COMMANDS	DROP VIEW	Java classes, resources, source code
NOAUDIT	LEAD	see ALTER commands		Materialized views
NOCOMPRESS	LEAST	ANALYZE	DB OBJECTS (NON SCHEMA)	Materialized view logs
NOT	LENGTH	ASSOCIATE STATISTICS	Contexts	Object tables
NOWAIT	LN	COMMENT	Directories	Object types
NULL	LOCALTIMESTAMP		PFILES & SPFILES	Object views
NUMBER	LOG	see CREATE commands	Profiles	Operators
OF	LOWER	DISASSOCIATE STATISTICS	Roles	Packages
OFFLINE	LPAD	see DROP commands	Rollback segments	Sequences
ON	LTRIM	EXPLAIN PLAN	Tablespaces	Stored functions, stored procedures
ONLINE	MAKE_REF		Users	Synonyms
OPTION	MAX			Tables
OR	MIN			Views
ORDER	MOD			

Fig. 1.3 Sheet of cheats Oracle

Reserved Words	Built-in Functions	Common Clauses/Misc Cmds	ALTER / DROP COMMANDS	CREATE Commands
PCTFREE	MONTHS_BETWEEN			
PRIOR	NCHR			
PRIVILEGES	NEW_TIME			SQL STANDARDS
PUBLIC	NEXT_DAY			ANSI Standards
RAW	NLS_CHARSET_DECL_LEN		SET OPERATORS	ISO Standards
RENAME	NLS_CHARSET_ID		UNION	FIPS Compliance
RESOURCE	NLS_CHARSET_NAME		UNION ALL	UNICODE
REVOKE	NLS_INITCAP		INTERSECT	Oracle Standards Compliance
ROW	NLS_LOWER		MINUS	
ROWID	NLSORT			American National Standards Institute
ROWNUM	NLS_UPPER			11 West 42nd Street
ROWS	NTILE			New York, NY 10036 USA
SELECT	NULLIF			Telephone: (212) 642-4600
SESSION	NUMTODSINTERVAL		JOIN OPERATORS	Fax: (212) 398-0023
SET	NUMTOYMINTERVAL		(+) - OUTER JOIN	http://ansi.org
SHARE	NVL		equijoins (=)	
SIZE	NVL2		INNER - inner join	
SMALLINT	PATH		RIGHT - right outer join.	International Organization for Standardization
START	PERCENT_RANK		LEFT - left outer join.	1 Rue de Varembe
			FULL - full or two-sided outer join	Case postale 56
			OUTER - join keyword following	
SUCCESSFUL	PERCENTILE_CONT		RIGHT, LEFT, or FULL	CH-1211, Geneva 20, Switzerland
SYNONYM	PERCENTILE_DISC		star joins	Telephone: +41 (22) 749-0111
SYSDATE	POWER			Fax: +41 (22) 733-3430
TABLE	RANK			http://www.iso.ch/
THEN	RATIO_TO_REPORT			
TO	RAWTOHEX			NIST / FIPS
TRIGGER	RAWTONHEX			http://www.it.nist.gov/fipspubs/
UID	REF			FIPS Std 127-2:
UNION	REFTOHEX			http://www.it.nist.gov/fipspubs/fip127-2.htm
UNIQUE	REGR (Linear Regression) Functions			
UPDATE	REPLACE			UNICODE
USER	ROUND (number)			http://www.unicode.org
VALIDATE	ROUND (date)			Oracle®i complies fully with Unicode 3.0
VALUES	ROW_NUMBER			
VARCHAR	ROWIDTOCHAR			
VARCHAR2	ROWIDTONCHAR			

Fig. 1.3 Sheet of cheats Oracle

Reserved Words	Built-in Functions	Common Clauses/Misc Cmds	ALTER / DROP COMMANDS	CREATE Commands
VIEW	RPAD			
WHENEVER	RTRIM			
WHERE	SESSIONTIMEZONE			
WITH	SIGN			
	SIN			
	SINH			
	SOUNDEX			
	SQRT			
	STDDEV			
	STDDEV_POP			
	STDDEV_SAMP			
	SUBSTR			
	SUM			
	SYS_CONNECT_BY_PATH			
	SYS_CONTEXT			
	SYS_DBURIGEN			
	SYS_EXTRACT_UTG			
	SYS_GUID			
	SYS_TYPEID			
	SYS_XMLAGG			
	SYS_XMLGEN			
	SYSDATE			
	SYSTIMESTAMP			
	TAN			
	TANH			
	TO_CHAR (character)			
	TO_CHAR (datetime)			
	TO_CHAR (number)			
	TO_CLOB			
	TO_DATE			
	TO_DSINTERVAL			
	TO_LOB			
	TO_MULTI_BYTE			
	TO_NCHAR (character)			
	TO_NCHAR (datetime)			
	TO_NCHAR (number)			
	TO_NCLOB			
	TO_NUMBER			
	TO_SINGLE_BYTE			
	TO_TIMESTAMP			
	TO_TIMESTAMP_TZ			
	TO_YMINTERVAL			

Fig. 1.3 Sheet of cheats Oracle

Reserved Words	Built-in Functions	Common Clauses/Misc Cmds	ALTER / DROP COMMANDS	CREATE Commands
	TRANSLATE			
	TRANSLATE ... USING			
	TREAT			
	TRIM			
	TRUNC (number)			
	TRUNC (date)			
	TZ_OFFSET			
	UID			
	UNISTR			
	UPDATEXML			
	UPPER			
	USER			
	USERENV			
	VALUE			
	VAR_POP			
	VAR_SAMP			
	VARIANCE			
	VSIZE			
	WIDTH_BUCKET			
	XMLAGG			
	XMLCOLATTVAL			
	XMLCONCAT			
	XMLELEMENT			
	XMLFOREST			
	XMLSEQUENCE			
	XMLTRANSFORM			
	ROUNDTRUNC Date Functions			
	User-Defined Functions			

Fig. 1.3 Sheet of cheats Oracle

CAPITULO 2 DISEÑO RELACIONAL DE BASE DE DATOS

2.1 Introducción

En general el objetivo del diseño de una base de datos relacional es generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información. Una de las técnicas para lograrlo consiste en diseñar esquemas que tengan una forma normal adecuada. Para determinar si un esquema de relaciones tiene una de las formas normales se requiere mayor información sobre la empresa del mundo real que se intenta modelar con la base de datos. La información adicional la proporciona una serie de limitantes que se denomina dependencias de los datos.

2.2 Proceso de desarrollo de base de datos

El desarrollo de base de datos en un enfoque *top-down*, que transforma los requerimientos de información en una base de datos operacional.

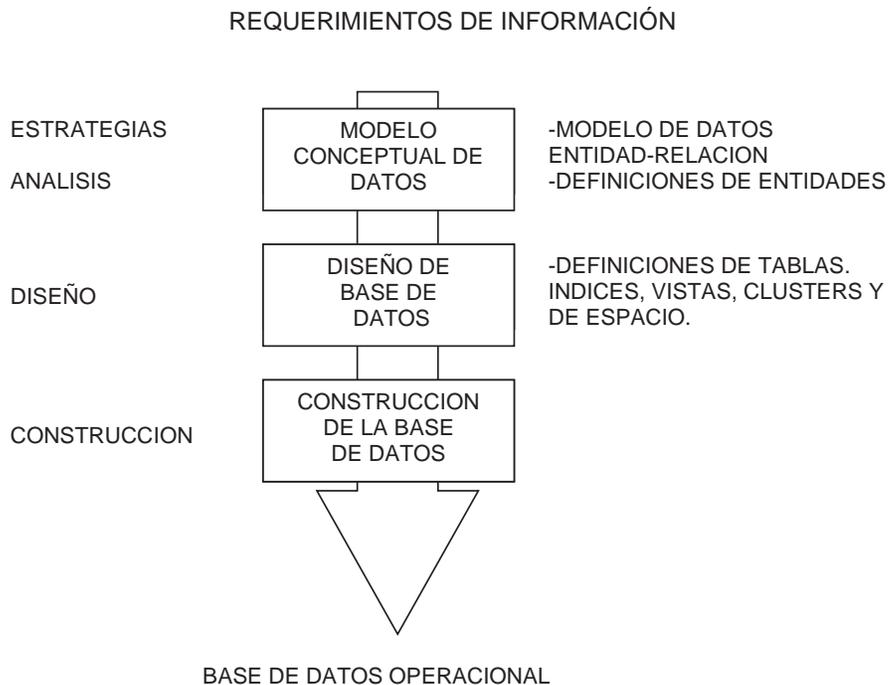


Fig. 2.1 Enfoque *top-down*

2.3 Requerimientos de información del negocio

El desarrollo *Top-Down* de la base de datos comienza con los requerimientos de información del negocio.

2.4 Introducción al modelo de datos conceptual

El modelo de datos conceptual define y modela los aspectos importantes a cerca de la información que el negocio necesita saber o tener y las relaciones entre dicha información.

Ejemplo:

El siguiente Modelo entidad relación representa los requerimientos del departamento de Recursos Humanos.

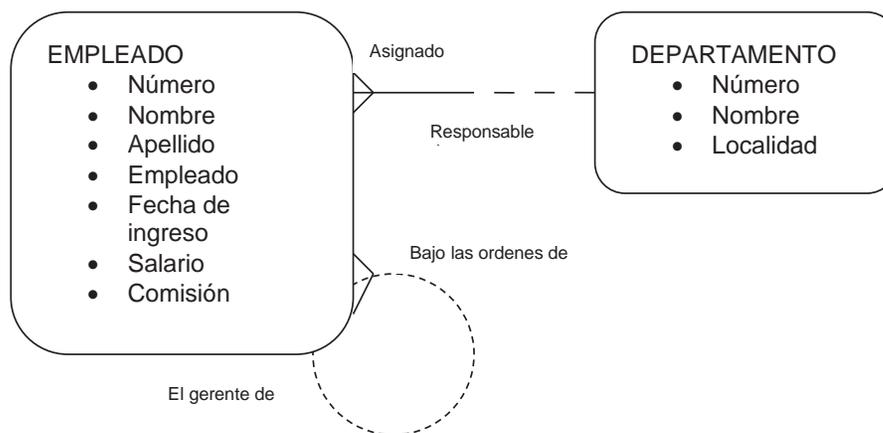


Fig. 2.2 Diagrama entidad relación

Un modelo de datos entidad relación debería modelar adecuadamente las necesidades de información de la organización y soportar las funciones del negocio.

En el diagrama del modelo entidad relación se encuentran las siguientes relaciones:

- Un empleado puede estar asignado solo a un departamento
- Un departamento puede ser responsable de uno o muchos empleados

- Además, cada empleado puede estar bajo las ordenes de uno y solo un empleado
- Pero, cada empleado puede ser gerente de uno o más empleados.

2.5 Introducción al diseño de base de datos

El diseño de base de datos, se mapean los requerimientos de información reflejados en un modelo entidad relación dentro de un diseño de base de datos relacional.

2.6 Introducción a la construcción de la base de datos

En la construcción de la base de datos, se crean físicamente las tablas en la base de datos relacional, implementándolas de acuerdo al diseño de la base de datos.

Las siguientes instrucciones de SQL, creará la tabla de nombre DEPARTAMENTO.

```
SQL> CREATE TABLE DEPARTAMENTO
2   NODEP  NUMBER (2) NOT NULL PRIMARY KEY,
3   NOMDEP CHAR (20)  NOT NULL,
4   LOC    CHAR (15)  NOT NULL;
```

Fig. 2.3 Instrucción SQL para crear una tabla.

El lenguaje SQL se usa para crear y manipular bases de datos relacionales.

2.7 Modelo conceptual de datos

El modelo conceptual de datos es el primer paso del proceso *Top-Down* para el desarrollo de base de datos, se ejecuta durante la fase de análisis y estrategia en el ciclo de desarrollo de sistemas.

El objetivo del modelo conceptual de datos es desarrollar el modelo entidad relación que representa los requerimientos de información de los negocios.

Componentes del modelo entidad-relación.

- Entidades. Son los aspectos importantes acerca de los cuales se necesita información.
- Relaciones. Como se relacionan las entidades.
- Atributos. Información específica la cual necesita ser almacenada.

Un modelo entidad relación es una forma efectiva para integrar y documentar los requerimientos de información de una organización.

Características.

- Un modelo entidad relación documenta los requerimientos de información de la organización en un formato preciso y claro
- Los usuarios pueden entender fácilmente la forma gráfica de un modelo entidad-relación.
- Un modelo entidad relación puede ser fácilmente desarrollado y refinado.
- Un modelo entidad relación provee una clara imagen del alcance de los requerimientos de información de las organizaciones.
- Un modelo entidad relación nos provee una estructura adecuada para la integración de múltiples aplicaciones, desarrollar proyectos, y/o paquetes de aplicación adquiridos.

2.8 Entidades

Una entidad es un aspecto importante acerca del cual se necesita tener o conocer información.

Los atributos describe entidades y son las piezas específicas de información las cuales necesitan ser conocidas.

Una entidad debe tener atributos que necesitan ser conocidos desde el punto de vista del negocio, de otra manera no es una entidad que forme parte del alcance de los requerimientos del negocio.

2.9 Relaciones

Una relación es bidireccional y representa la asociación entre dos entidades, o entre una entidad consigo misma.

Existen tres grados de relación.

- Relación de muchos a uno (M : 1)
- Relaciones de muchos a muchos (M : M)
- Relaciones de uno a uno (1 : 1)

2.9.1 Relación muchos a uno (M : 1)

- Tiene un grado de uno o más en una parte de la relación y de uno y solo uno en la otra parte.
- Es el tipo de relación más común dentro de las bases de datos.
- Las relaciones de muchos a uno que sea obligatoria en ambas partes es rara.

2.9.2 Relación muchos a muchos (M : M)

- Tiene un grado de uno o más en ambas partes.
- También es un tipo de relación común.
- Pueden ser opcionales en una o en ambas partes.

2.9.3 Relación uno a uno (1 : 1)

- Tiene un grado de uno y sólo uno en ambas partes.
- Este tipo de relación es raro y más aún si ambas partes son obligatorias.

- Este tipo de relación podría indicar que ambas relaciones se puedan convertir en solo una.
-

Todas las relaciones deben representar los requerimientos de información y reglas del negocio.

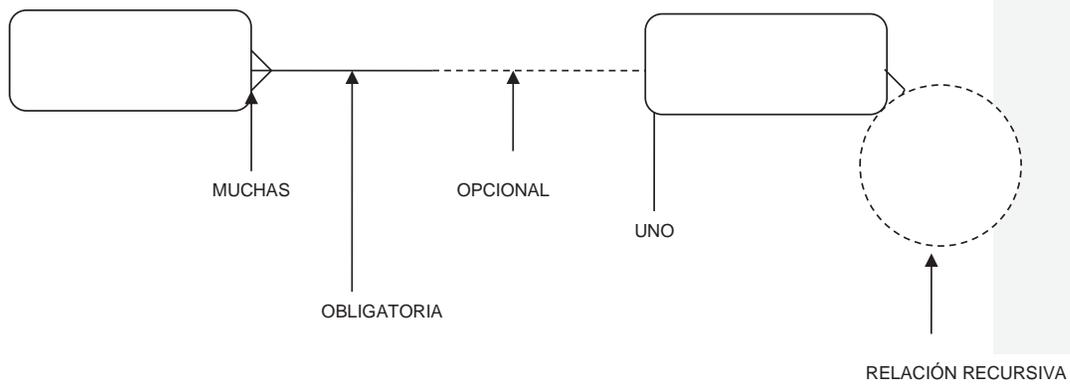


Fig. 2.4 Estándares de diagramación.

2.10 Matriz de relaciones

Las matrices de relaciones se usan como ayuda para la colección inicial de información sobre las relaciones entre una serie de entidades.

Estándares de matriz de relaciones.

- Una matriz de relaciones muestra si están relacionadas y en qué forma cada entidad (renglón) con cada entidad (columna) mostrada en la matriz.
- Todas las entidades están listadas en el lado izquierdo y en la parte superior de la matriz.
- Si una entidad no está relacionada con otra entidad, entonces se dibuja una línea en la caja de intersección.

- Cada relación por encima de la diagonal es el inverso o la imagen espejo de la relación por debajo de la línea diagonal.
- Las relaciones recursivas (una entidad consigo misma) son representadas por las cajas en la diagonal.

	FACILIDAD	SLIP	SOLICITUD	CLIENTE	SERVICIO
FACILIDAD	_____	ubicar	_____	_____	_____
SLIP	estar ubicado en	_____	crear	rentado por	_____
SOLICITUD	_____	creada por	_____	_____	contener
CLIENTE	_____	rentar	_____	_____	_____
SERVICIO	_____	_____	contenido en	_____	_____

Fig. 2.5 Matriz de relaciones.

Detalles importantes al utilizar las matrices de relaciones

- Relaciona las entidades de la parte izquierda con las entidades de la parte derecha.
- Deben ponerse todas las entidades y en el mismo orden en ambos lados.
- Al encontrar dos entidades que se relacionen, se pone el nombre de la relación, de no relacionarse, se pone una raya.
- La línea del medio divide y crea un efecto de espejo entre las relaciones.

2.11 Analizar y modelar relaciones

Seguir una serie de cinco pasos para analizar y modelar relaciones.

Pasos

1. Determinar si existe una relación.

2. Nombrar cada dirección de la relación.
3. Determinar la opcionalidad de cada dirección de la relación.
4. Determinar el grado de cada dirección de la relación
5. Leer en voz alta las relaciones para validarlas.

2.11.1 Determinar si existe una relación.

Examinar en cada par de entidades para determinar si existe una relación. Cuestionar si existe una relación.

- ¿Existe una relación significativa entre la entidad A y la entidad B?

2.11.2 Nombrar cada relación.

Cuestionar en nombre adecuado para la relación.

- ¿Cómo está relacionada la entidad A con la entidad B?
Una entidad A es nombre de la relación de una entidad B.
- ¿Cómo está relacionada la entidad B con la entidad A?
Una entidad B es nombre de la relación de una entidad A

Opcionalmente, registrar el nombre de la relaciones dentro de la matriz.

Usar una lista de pares de nombres de relaciones para ayudar a ponerle nombre a dichas relaciones.

Pares de nombres para las relaciones.

- Basado en la base para
- Cliente de el proveedor de
- Descripción de para
- Operador por el operador de
- Representado por la representación de
- Responsable de las responsabilidad de

Es importante mencionar que los nombres como *relacionado a* o *asociado con* no se pueden usar.

2.11.3 Determinar la opcionalidad de una relación

Es cuestionar acerca de una relación opcional.

- ¿Debe la entidad A ser nombre de la relación de la entidad B?
- ¿Debe la entidad B ser nombre de la relación de la entidad A?

2.11.4 Determinar el grado de relación

Determinar el grado de relación de ambas direcciones.

Cuestionar el grado de relación.

- ¿Puede la entidad A ser nombre de la relación de más de una de la entidad B?
- ¿puede la entidad B ser nombre de la relación de más de una de la entidad A?

2.11.5 Validar la relación

Volver a examinar el modelo entidad-relación y validar la relación.

Leer en voz alta la relación.

- Las relaciones deben ser fáciles de leer y tener sentido en el negocio.

2.12 Representación del diagrama entidad-relación

Hacer un diagrama entidad-relación fácil de leer y aplicarlo para la gente que necesita trabajar con él.

Limpio y ordenado.

- Alinear las cajas de las entidades.
- Dibujar las líneas de la relación como rectas horizontales o verticales

- Usar un ángulo de 30° a 60° grados el cual facilita seguir las líneas de la relación cuando estas se cruzan
- Evitar el uso de muchas líneas paralelas ya que se dificulta el seguimiento.

Texto claro.

- Hacer todo el texto claro.
- Evitar abreviaciones y modismos
- Agregar adjetivos para mejorar el entendimiento
- Alinear el texto horizontalmente
- Poner el nombre de la relación final de la línea y lados opuestos de la línea.

Formas fáciles de recordar

- Hacer el diagrama entidad-relación fácil de recordar. Que la gente recuerde las formas y los patrones.
- No se debe dibujar diagramas entidad-relación en una cuadrícula.
- Compactar en la medida de lo posible las cajas de las entidades para ayudar a la visualización del diagrama.

2.13 Atributos

Los atributos son información que se necesita conocer o tener a cerca de una entidad. Los atributos describen una entidad para calificar, identificar, clasificar, cuantificar o expresar el estado de una entidad.

Los atributos representan un tipo de descripción o detalle, más no una instancia.

2.13.1 Puntos importantes a tomar en cuenta sobre los atributos

- Los nombres de los atributos están en singular y se muestran en minúsculas.
- Todos los atributos se deben descomponer hasta su mínimo componente con significado.

- Se debe verificar que cada atributo tenga un solo valor para cada instancia. Los atributos multivalor o un grupo de repetición no es un atributo válido.
- Verificar que un atributo no sea derivado o calculado de los valores existentes de otros atributos.

2.14 Identificadores únicos

Un identificador único (UID) es cualquier combinación de atributos y/o relaciones que sirven para identificar en forma única una ocurrencia o instancia de una entidad. Cada ocurrencia de una entidad debe ser identificada de una manera única.²

2.15 Normalizar el modelo de datos

Normalizar es un concepto de base de datos relacional, pero sus principios se aplican al modelo conceptual de datos.

2.15.1 Reglas de normalización

Primera forma normal (1FN)³

Todos los atributos deben tener un solo valor para cada instancia

Segunda forma normal (2FN)⁴

Un atributo debe ser dependiente del identificador.

Tercera forma normal (3FN)⁵

Ningún atributo no-UID puede ser dependiente de otro atributo no-UID.⁶

² UID: Unique Identifier (Identificador único).

³ 1FN: Primera forma normal.

⁴ 2FN: Segunda forma normal.

⁵ 3FN: Tercera forma normal.

⁶ No-UID: Not unique identifier (No identificador único).

Un modelo de datos entidad-relación normalizado se traslada automáticamente dentro de un diseño de base de datos.

2.15.1.1 Regla de la primera forma normal

Todos los atributos deben tener un solo valor para cada instancia. Validar que cada atributo tenga un valor único para cada ocurrencia. Ningún atributo deberá tener valores repetidos.

Si un atributo tiene múltiples valores, sea crea una entidad adicional y lo relaciona con la entidad original mediante una relación M:1⁷.

2.15.1.2 Regla de la segunda forma normal.

Un atributo debe ser independiente del identificador único completo. Validar que cada atributo dependa completamente del UID. Cada instancia específica del UID debe determinar una sola instancia de cada atributo, otro punto que se debe validar es que un atributo dependa de una sola parte del UID de la entidad. Si un atributo no es dependiente del UID completo, está fuera de lugar y deberá ser movido.⁸

2.15.1.3 Regla de la tercera forma normal

Ningún atributo no-UID puede ser dependiente de otro atributo no-UID. Además se debe validar que cada tributo no-UID no dependa de otro atributo no-UID. Se debe mover cualquier atributo no-UID que dependa de otro atributo no-UID.

Si un atributo depende de otro atributo no-UID, es necesario mover ambos, el atributo dependiente y el atributo del que depende, a una nueva entidad relacionada con la entidad actual.⁹

2.16 Introducción a bases de datos relacionales

⁷ M:1: Relación muchos a uno.

⁸ UID: Unique identifier (Identificador único).

⁹ No-UID: Not unique identifier (No identificador único).

Una base de datos relacional es una base de datos que es percibida por el usuario como una colección de relaciones de tablas de dos dimensiones.

- Las tablas de base de datos relacional son sencillas pero disciplinarias.
- Una base de datos relacional debe tener integridad de datos, sus datos deben ser precisos y consistentes.

Las bases de datos relacionales son manipuladas como un conjunto en un tiempo en vez de registro en un tiempo.

- El Lenguaje Estructurado de Consulta (*Structured Query Language (SQL)*) es utilizado para manipular las bases de datos relacionales.
- El Instituto Nacional Americano de Estándares (*ANSI*) ha establecido a *SQL* como el lenguaje estándar para operar sobre las bases de datos relacionales.¹⁰
- Una base de datos relacional puede soportar un conjunto completo de operaciones relacionales. Las operaciones relacionales manipulan conjunto de valores de datos. Las tablas pueden ser utilizadas en la creación de otras tablas. Las operaciones relacionales pueden ser anidadas.

2.17 Llaves primarias

Una llave primaria (PK) es una columna o grupo de columnas que identifican de manera única a cada renglón en una tabla. Cada tabla debe tener una llave primaria y una llave primaria debe ser única.¹¹

- No se aceptan duplicados en una llave primaria. La llave primaria debe ser única.
- El valor de las llaves primarias generalmente no pueden cambiar.
- El UID de una entidad irá de acuerdo con la llave primaria en su tabla.¹²

Una llave primaria que consta de múltiples columnas se llama llave primaria compuesta.

¹⁰ ANSI: American National Standards Institute.

¹¹ PK: Primary Key (Llave primaria).

¹² UID: Unique identifier (Identificador único).

- Las columnas de una llave primaria compuesta deben ser únicas en combinación. Las columnas pueden tener duplicados en forma individual, pero en combinación, no se permiten duplicados.

Ninguna parte de la llave primaria puede ser nula.

Una tabla puede tener más de una columna o combinación de columnas que pueden servir como la llave primaria de la tabla. Cada una de estas es llamada llave candidata o alterna.

- Todas las llaves alternas deben ser únicas y no nulas.
- Los UID secundarios concuerdan con las llaves alternas.
- Los nombres de personas normalmente no son llaves alternas por que no se puede garantizar que sean únicas.

2.18 Llaves foráneas

Una llave foránea (FK) es una columna o combinación de columnas en una tabla, que se refieren a una llave primaria en la misma o en otra.¹³

- Las llaves foráneas son utilizadas por hacer *JOIN* entre tablas.
- Las llaves foráneas se basan en los valores de los datos y son puramente lógicas.
- La llave foránea puede ser repetida

Una llave foránea debe coincidir con un valor de una llave primaria existente. Si la llave foránea es parte de una llave primaria, la llave foránea no puede ser nula.

2.19 Integridad de datos

La integridad de datos se refiere a la exactitud y consistencia de los datos.

2.19.1 Constraints de integridad de datos

- Los constraints de integridad de datos definen al estado relacional correcto de la base de datos.

¹³ FK: Foreign Key (Llave foránea).

- Los constraints de integridad de datos aseguran que los usuarios realizarán únicamente operaciones en las cuales dejarán a la base de datos en un estado correcto y consistente.

2.19.1.1 Integridad de entidades

Ninguna parte de la llave primaria puede ser nula.

2.19.1.2 Integridad referencial

Una llave foránea debe coincidir con un valor de una llave primaria

2.19.1.3 integridad de columnas

Una columna debe contener sólo valores consistentes con el formato de datos definido para la columna.

2.19.1.4 Integridad definida por el usuario

Los datos almacenados en la base de datos deben cumplir con las reglas del negocio.

Todos los *constraints* de integridad de datos deben ser reforzados por el DBMS (*Database Management System*) o el software de aplicación.

- Un dato es inconsistente si existen múltiples copias de un registro y no todas las copias han sido actualizadas. Una base de datos inconsistente puede proveer información incorrecta o contradictoria a los usuarios.

Las reglas de negocio también pueden determinar el estado correcto de una base de datos. Estas reglas de negocio son llamados constraint de integridad de datos definidos por el usuario.

- Los constraints de los datos definidos por el usuario pueden ser administrados por políticas o ser requeridos por las leyes gubernamentales.

- Frecuentemente esas reglas son completamente arbitrarias o al menos parecen ser arbitrarias.
- Los constraints de integridad de datos definidos por el usuario pueden incluir múltiples columnas y tablas.

2.20 Diseño de la base de datos

El diseño de la base de datos es ejecutado durante la etapa de diseño del ciclo de desarrollo del sistema y es ejecutado conjuntamente con el diseño de aplicaciones.

El diseño de la base de datos se lleva a cabo por medio de dos actividades.

1. pasar el modelo entidad-relación a tablas relacionales para producir el diseño inicial.
2. refinar el diseño inicial para producir un diseño completo de la base de datos.

2.20.1 Liberación del diseño de la base de datos

La etapa de diseño de la base de datos produce especificaciones de diseño para una base de datos relacional, incluyendo definiciones para tablas relacionales, índices, vistas y espacio de almacenamiento.

2.20.2 Mapear entidades

Mapear la tabla para cada entidad. Crear un mapa de instancias para la nueva tabla. Registrar únicamente el nombre de la tabla.

- El nombre de la tabla debe ser fácil de identificar con el nombre de la entidad. El nombre en plural de una entidad se usa algunas veces porque la tabla debe contener un grupo de renglones.

2.20.3 Mapear atributos a columnas

Mapear cada atributo de la entidad a una columna en su tabla correspondiente. Establecer los atributos obligatorios para columnas (no nulas).

- El nombre de las columnas debe ser fácil de identificar un modelo entidad-relación.
- Usar abreviaciones consistentes que no causen confusión al usuario y al programador.
- Los nombres de las columnas cortos o pequeños reducirán el tiempo requerido para el comando de SQL “*parsing*”.¹⁴

2.20.4 Mapear UIDS (Identificador Único) a llaves primarias

Asignar cualquier atributo(s) que sea parte del UID de la entidad a columnas PK. Etiquetar las columnas PK.^{15 16}

- Todas las columnas etiquetadas con PK deben etiquetarse también con NN y U
- Asignar un UID que incluya atributos a una PK compuesta. Etiquetar estas columnas NN y UID.¹⁷
- Si una entidad incluye una relación, agregar columnas de llaves foráneas para la tabla y señalarlas como parte de la llave primaria.

2.20.5 Mapear relaciones para llaves foráneas

Para una relación de entidades M:1 (muchos a uno), se debe tomar el PK de la tabla(1) y ponerlo en la tabla (M).¹⁸

- Elegir un nombre único para la columna FK y etiquetar la(s) columna(s) FK.¹⁹

2.21 Conclusiones

Como parte fundamental de una base de datos es que se tenga una buena relación es por eso que el modelo relacional de bases de datos con sus relaciones normalizadas es una solución simple para satisfacer las más diversas condiciones de consulta y extracción de datos e información.

¹⁴ SQL: Structured Query Language (Lenguaje de consulta estructurado).

¹⁵ UID: Unique Identifier (Identificador único).

¹⁶ PK: Primary Key (Llave primaria).

¹⁷ NN: Not Null (No nulo).

¹⁸ PK: Primary Key (Llave primaria).

¹⁹ FK: Foreign Key (Llave foránea).

CAPITULO 3 REVISIÓN TÉCNICA

3.1 Introducción

Es muy importante mencionar y/o mostrar el proceso mediante el cual fue llevado a cabo este proyecto de tal manera que se pueda permitir que el fundamento teórico sustente el desarrollo del proyecto.

De esta manera se tomará este capítulo con el objetivo de explicar lo que se llevó a cabo durante el proceso del proyecto.

3.2 Instalación del SQL SERVER 2005

Como se vio en el capítulo 2 fue importante tener una capacitación de SQL SERVER con la finalidad de comprender mejor lo que se tenía como base del sistema de recursos humanos y de esta manera poder tener una mejor idea del proyecto que se tenía en puerta, además de tener grandes posibilidades de éxito.

Durante esta etapa el personal encargado de redes y soporte se encargó de instalar en los equipos del personal de desarrollo y migración el SQL SERVER 2005 con el fin de tener la base de datos de la empresa que solicitó la migración del sistema de recursos humanos.

Para la instalación del SQL SERVER 2005 se tuvieron que tener en cuenta los siguientes 3 pasos:

1. Preparar el equipo para instalar el SQL SERVER 2005
2. Instalar SQL SERVER 2005.
3. Configuración de la instalación del SQL SERVER 2005.

3.2.1 Preparar el equipo para instalar SQL SERVER 2005

Para preparar el equipo para SQL SERVER 2005, se tuvo que revisar los requisitos del *hardware* y *software*, los requisitos del comprobador de configuración del sistema, los problemas de bloqueo y las consideraciones de seguridad.

3.2.2 Instalar SQL SERVER 2005

Para instalar SQL SERVER 2005, es necesario que se ejecute el programa de instalación mediante el asistente para la instalación de SQL SERVER 2005 o realizar la instalación desde el símbolo del sistema.

3.2.3 Configuración de la instalación del SQL SERVER 2005

Después de que el programa de instalación complete la instalación de SQL SERVER 2005, se puede configurar SQL SERVER mediante utilidades gráficas o del símbolo del sistema.

3.3 Instalación de Oracle developer

En esta etapa al igual que en la anterior se necesitó realizar una instalación del Oracle Developer, ya que aquí fue donde se realizó toda la migración, para ello es importante tener en cuenta los siguientes puntos:

1. Primero se tiene que comprobar en el panel de control que esté instalado el protocolo TCP/IP
2. Una vez que se tenga el software, se ejecuta y aparecerá el asistente de instalación dando la bienvenida



Fig. 3.1 Inicio instalación Oracle developer

3. A continuación aparecerá la pantalla de ubicación de ficheros de origen y destino



Fig. 3.2 Ubicación de archivos

4. Después se mostrara la pantalla para que se seleccione el tipo de instalación, se debe seleccionar la que más nos convenga.
5. En este punto preguntará el tipo de configuración de la base de datos según el uso que se le dará, se selecciona uso general se pulsa siguiente.

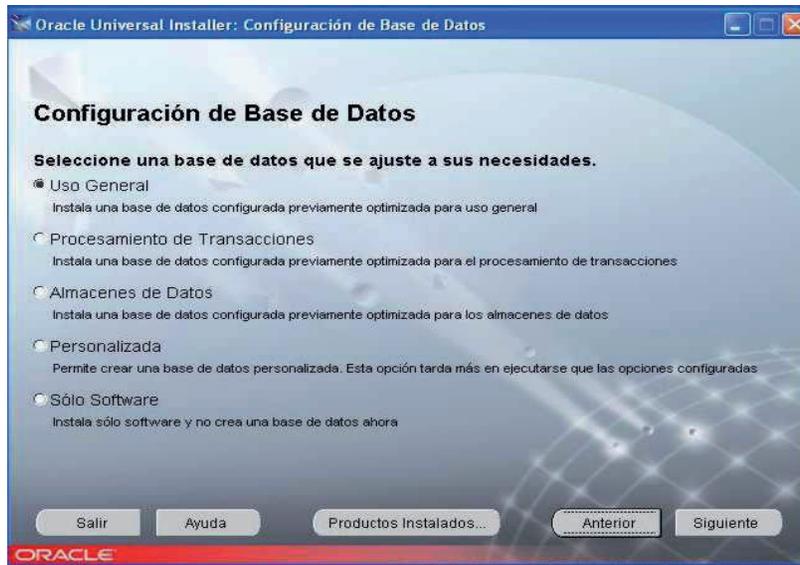


Fig. 3.3 Configuración de la base de datos

6. Ahora preguntará por el puerto a usar por Oracle MTS *Recovery Service*, se debe dejar por defecto (2030) y se pulsa el botón siguiente



Fig. 3.4 Oracle Services

- Este punto es importante, ya que se pide la ubicación de los archivos de datos, ahí se debe dejar el valor por defecto.



Fig. 3.5 Ubicación de archivos

- Finalmente aparece un pequeño resumen con distintas opciones de la instalación, se pulsa el botón instalar.

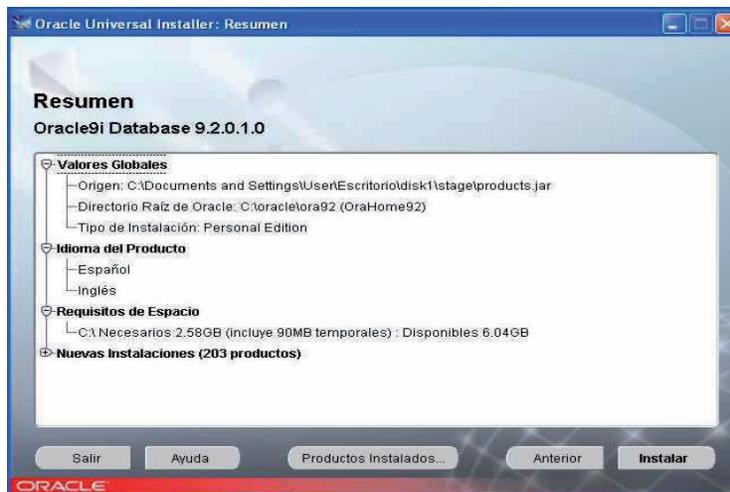


Fig. 3.6 Resumen

- Después de un rato de instalación y configuración, pedirá las contraseñas para los superusuarios de la base de datos (*SYS* Y *SYSTEM*).

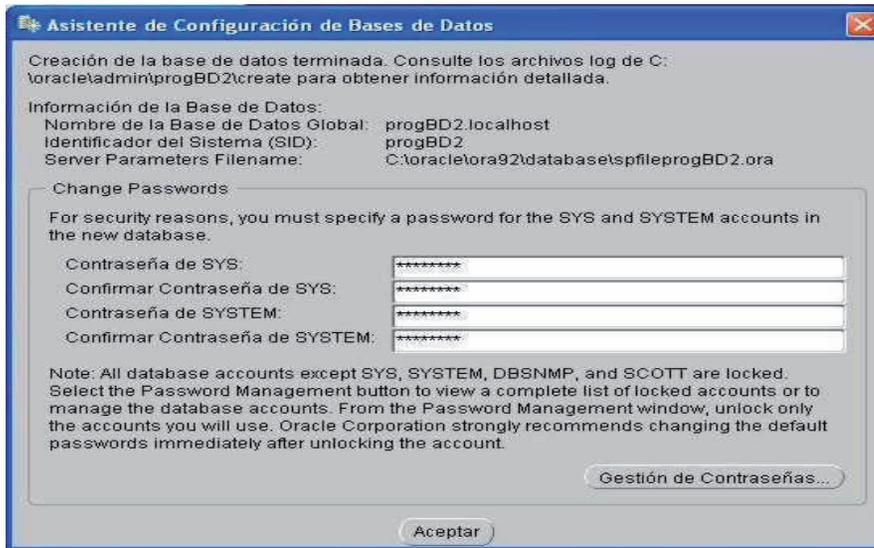


Fig. 3.7 Asistente de configuración

- Si no surgió ningún error aparecerá la pantalla de fin de la instalación.

En este punto fue de mucha importancia hacer mención de cómo se instaló el Developer, ya que como se mencionó con anterioridad aquí fue donde se realizó la mayor parte de la migración del sistema.

3.4 Capacitación

Para comprender aún más el proceso que se tenía que llevar a cabo con respecto a la migración fue necesario recibir una capacitación Express de un mes con el objetivo principal de aprender a utilizar las herramientas que anteriormente se mencionaron, SQL SERVER, Oracle y *Visual Basic* 6.0 para llevar a cabo la migración.

A continuación se muestra los temas de capacitación, es importante mencionar que solo se mostrarán los temas ya que la explicación de los mismos ya se hizo en los capítulos anteriores.

3.4.1 Capacitación diseño relacional de base de datos

- Proceso de desarrollo de base de datos
- Requerimientos de información del negocio
- Introducción al modelo de datos conceptual
- Introducción al diseño de base de datos
- Introducción a la construcción de la base de datos
- Modelo conceptual de datos
- Entidades
- Relaciones
- Relación muchos a uno (M : 1)
- Relación muchos a muchos (M : M)
- Relación uno a uno (1 : 1)
- Matriz de relaciones
- Analizar y modelar relaciones
- Determinar si existe una relación.
- Nombrar cada relación.
- Determinar la opcionalidad de una relación
- Determinar el grado de relación
- Validar la relación
- Representación del diagrama entidad-relación
- Atributos
- Puntos importantes a tomar en cuenta sobre los atributos
- Identificadores únicos
- Normalizar el modelo de datos
- Reglas de normalización
- Regla de la primera forma normal
- Regla de la segunda forma normal
- Regla de la tercera forma normal
- Introducción a bases de datos relacionales
- Llaves primarias

- Llaves foráneas
- Integridad de datos
- Constraints de integridad de datos
- Integridad de entidades
- Integridad referencial
- integridad de columnas
- Integridad definida por el usuario
- Diseño de la base de datos
- Liberación del diseño de la base de datos
- Mapear entidades
- Mapear atributos a columnas
- Mapear UIDS a llaves primarias
- Mapear relaciones para llaves foráneas

3.4.2 Capacitación SQL SERVER 2005

- Conceptos básicos sobre bases de datos relacionales
- ¿Qué es SQL?
- Características del lenguaje
- Cómo interpretar un diagrama sintáctico
- Consultas Simples
- Tabla Origen From
- Selección de columnas
- Columnas de la tabla origen
- Columnas calculadas
- Ordenación de las filas-*ORDER BY*
- Cláusula *DISTINCT / ALL*
- Cláusula *TOP*
- Cláusula *WHERE*
- Condiciones de selección
- Test de comparación
- Test de rango (*BETWEEN*)
- Test de pertenencia a conjunto (*IN*)

- Test de valor nulo (*IS NULL*)
- Test de correspondencia con patrón (*LIKE*)
- Consultas multitabla
- Unión de tablas
- Composición de tablas
- Operador *UNION*
- Producto cartesiano
- *INNER JOIN*
- *LEFT JOIN* Y *RIGHT JOIN*
- Consultas de resumen
- Funciones de columna
- Cláusula *GROUP BY*
- Cláusula *HAVING*
- Subconsultas
- Referencias externas
- Anidar subconsultas
- Subconsulta en la lista de selección
- Subconsultas en la cláusula *FROM*
- Subconsulta en las cláusulas *WHERE* y *HAVING*
- Condiciones de selección con subconsultas
- El *test* de comparación con subconsulta
- El *test* de comparación cuantificada
- *Test ANY*
- *Test ALL*
- *Test* de pertenencia a conjunto (*IN*)
- *Test* de existencia *EXISTS*
- Insertar una fila (*INSERT INTO VALUES*)
- Inserta varias filas *INSERT INTO SELECT*
- Insertar filas en una nueva tabla *SELECT INTO*
- Actualización de datos (*UPDATE*)
- Borrar filas (*DELETE*)

3.4.3 Capacitación *Visual Basic 6.0*

- Tipos de datos
- Convertir tipos de datos
- Variables
- Almacenamiento y recuperación de datos en variables
- Declaración de variables
- Declaración implícita
- Declaración explícita
- Alcance de las variables
- Constantes
- Creación de constantes propias
- Alcance de las constantes definidas por el usuario
- Arreglo de variables
- Arreglos dinámicos
- Estructuras de control
- Estructuras de decisión
- *If...Then*
- *If...Then...Else*
- *Select Case*
- Estructuras de repetición (bucle)
- *Do...Loop*
- *For...Next*
- *For Each...Next*
- Salida de una estructura de control
- Funciones
- Funciones privadas
- El entorno integrado de desarrollo (IDE)
- Formulario
- Controles básicos
- Control Etiqueta (*Label*)

- Cuadro de texto (*TextBox*)
- Botón de comando (*Commandbutton*)
- marco (*Frame*)
- casilla de verificación (*CheckBox*)
- Botón de opción (*OptionButton*)
- Barra de Desplazamiento Horizontal (*HScrollBar*) y Barra de Desplazamiento Vertical (*VScrollBar*)
- Cuadro combinado (*ComboBox*)
- Cuadro de lista de unidades (*DriveListBox*)
- Cuadro de Lista de directorios (*DirListBox*)
- Cuadro de lista de archivos (*FileListBox*)
- Imagen (*Image*)
- Temporizador (*Timer*)
- *Data*
- Motor de base de datos *Microsoft jet*
- Base de datos *jet*
- Bases de datos método de acceso secuencial indexado (ISAM)
- Bases de datos compatibles con ODBC (*Open DataBase Connectivity- Conectividad Abierta de Base de Datos*)
- Otros métodos de acceso a datos
- Otros métodos de acceso a datos
- Librerías ODBC
- Librerías SQL de *visual Basic*
- Acceso a datos con el control *Data*
- Uso de controles enlazados a datos
- Establecer las propiedades del control data
- Enlazar controles
- Propiedades y métodos del control data
- Propiedad *Connect*
- Propiedad *Exclusive*
- Propiedad *Recordset*
- Propiedades *BOFAction* y *EOFAction*
- Método *Refresh*

- Objeto *Recordset*
- ¿Qué es un *Recordset*?
- Determinar los límites de un *Recordset*
- Uso de las propiedades y métodos de un *Recordset*
- Propiedades *BOF* y *EOF*
- Método *AddNew* del *Recordset*
- *UpdateRecord* del control *Data*
- Método *CancelUpdate* del control *Data*
- Método *Delete* del *Recordset*
- Eventos del control *Data*
- Evento *Validate*
- Argumento *Action*
- Argumento *Save*
- *DBGrid*
- *MSFlexGrid*
- *DBCombo*

3.4.4 Capacitación de Oracle

- Tipos de datos soportados por ORACLE
- Tipo de dato *CHAR()*
- Tipo de dato *VARCHAR2()*
- Tipo de dato *NCHAR()*
- Tipo de dato *NVARCHAR2()*
- Tipo de dato *NUMBER(p,s)*
- Tipo de dato *FLOAT(b)*
- Tipo de dato *DATE*
- Tipos de datos binarios
- Tipo de dato *LONG*
- Tipo de dato *ROWID*
- Crear tablas
- Ingresar registros

- Recuperación de algunos campos (*select*)
- Recuperación algunos registros (*where*)
- Operadores relacionales
- Actualizar registros (*update*)
- Comentarios
- Operadores relacionales (*is null*)
- Clave primaria (*primary key*)
- Vaciar la tabla (*truncate table*)
- Ingresar algunos campos
- Alias (encabezados de columnas)
- Funciones matemáticas
- Funciones de fechas y horas
- Ordenar registros (*order by*)
- Otros operadores relacionales (*between*)
- Operador relacional (*in*)
- Búsqueda de patrones (*like - not like*)
- Agrupar registros (*group by*)
- Seleccionar grupos (*Having*)
- Clave primaria compuesta
- Secuencias (*create sequence - currval - nextval - drop sequence*)
- Alterar secuencia (*alter sequence*)
- Integridad de datos
- Restricción *primary key*
- Restricción *unique*
- Restricción *check*
- Restricciones: validación y estados (*validate - novalidate - enable - disable*)
- Restricciones: información (*user_constraints - user_cons_columns*)
- Restricciones: eliminación (*alter table - drop constraint*)
- Índices
- Índices (Crear - Información)
- Índices (eliminar)
- Varias tablas (*join*)
- Combinación interna (*join*)

- Combinación externa izquierda (*left join*)
- Combinación externa derecha (*right join*)
- Combinación externa completa (*full join*)
- Combinaciones cruzadas (*cross*)
- Auto combinación
- Combinaciones y funciones de agrupamiento
- Combinar más de 2 tablas
- Clave foránea
- Restricciones (*foreign key*)
- Restricciones *foreign key* en la misma tabla
- Restricciones *foreign key* (eliminación)
- Restricciones *foreign key* deshabilitar y validar
- Restricciones *foreign key* (acciones)
- Restricciones al crear la tabla
- Unión
- Intersección
- *Minus*
- Agregar campos (*alter table- add*)
- Modificar campos (*alter table- modify*)
- Eliminar campos (*alter table- drop*)
- Subconsultas
- Subconsultas como expresión
- Subconsultas con in
- Subconsultas any- some- all
- Subconsultas correlacionadas
- *Exists* y *No Exists*
- Subconsulta con *update* , *delete* e *insert*
- Vistas (*create view*)
- Vistas eliminar (*drop view*)
- Vistas (modificar datos a través de ella)
- Vistas (*with read only*)
- Vistas modificar (*create or replace view*)
- Vistas (*with check option*)

- Procedimientos almacenados
- Crear y ejecutar un procedimiento almacenado
- Eliminar un procedimiento almacenado
- Parámetros de entrada de un procedimiento almacenado
- Variables en los procedimientos almacenados
- Funciones
- Control de flujo (*if*)
- Control de flujo (*case*)
- Control de flujo (*loop*)
- Control de flujo (*for*)
- Control de flujo (*while loop*)
- Disparador (*trigger*)
- Disparador de inserción a nivel de sentencia
- Disparador de inserción a nivel de fila (*insert trigger for each row*)
- Disparador de borrado (nivel de sentencia y de fila)
- Disparador de actualización a nivel de sentencia (*update trigger*)
- Disparador de actualización a nivel de fila (*update trigger*)
- Disparador de actualización - lista de campos (*update trigger*)
- Disparador de múltiples eventos
- Disparadores (habilitar y deshabilitar)
- Disparador (eliminar)

3.5 Configuración del origen de datos (ODBC)

En este punto se muestra la manera en que se configura el origen de datos tomando en cuenta primeramente el nombre del origen de datos y el controlador del mismo, a continuación se muestra gráficamente.



Fig. 3.8 Configuración ODBC

Seleccionar el driver de Oracle de la lista disponible

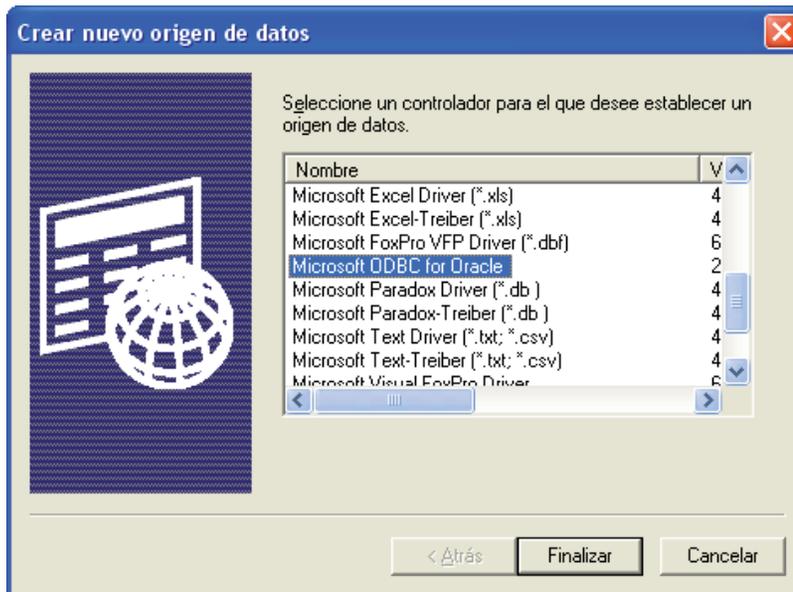


Fig. 3.9 Creación de nuevo origen de datos

Posteriormente se indica el nombre del origen de datos, descripción, nombre del servidor y la versión del cliente como se muestra.

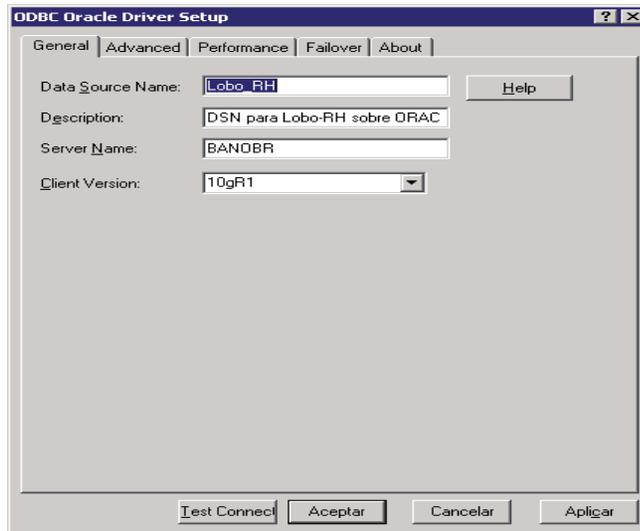


Fig. 3.10 Driver

Una vez ingresado todos los parámetros se da clic sobre el botón test conexión para verificar que este bien hecha la conexión y finalmente se pulsa aceptar.

3.6 Configuración del SQL Developer

En este punto se hablará sobre la configuración en el programa SQL Developer para dar de alta una nueva conexión hacia la base de datos, que fue el programa que se utilizó para la migración.

A continuación se mencionan los pasos para realizar dicha tarea:

Paso 1: Abrir el SQL Developer y en el icono de Connections hacer clic derecho y en el texto "New Connection".

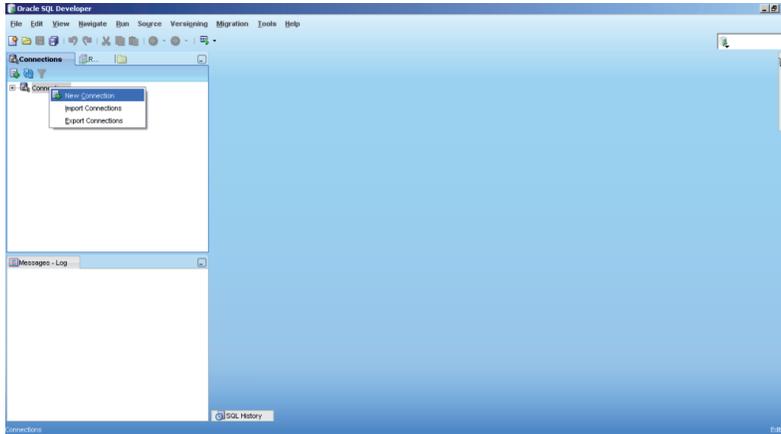


Fig. 3.11 Developer

Paso 2: Configuración del origen de la base de datos.

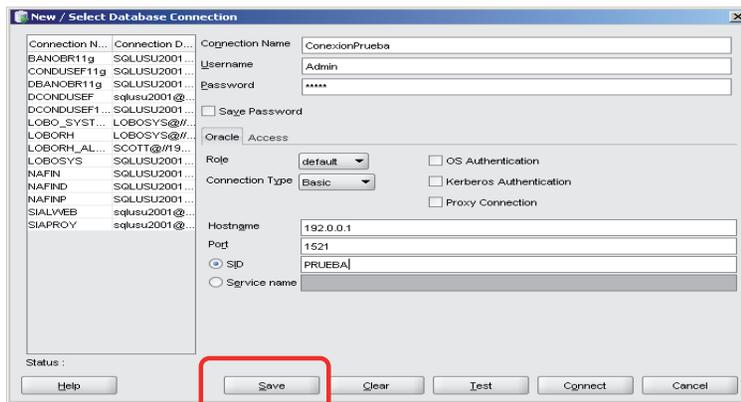


Fig. 3.12 Configuración de la base de datos

Dónde:

Connection Name: Es el nombre con el cual aparecerá la base de datos en SQL Developer.

UserName: Es la clave de usuario con la cual se conecta a la base de datos.

Password: Es la clave de la contraseña con la cual se conecta a la base de datos.

Hostname: Es la IP del servidor donde se encuentra la base de datos. En caso de estar creando la conexión desde el servidor se pone "Localhost".

Port: Es el puerto de salida de la base de datos en el servidor.

SID: Es el nombre del servicio de la base de datos de Oracle configurado en el servidor y finalmente hacer clic en Save.

Paso 3: Se muestra la interfaz una vez conectado a la base de datos.

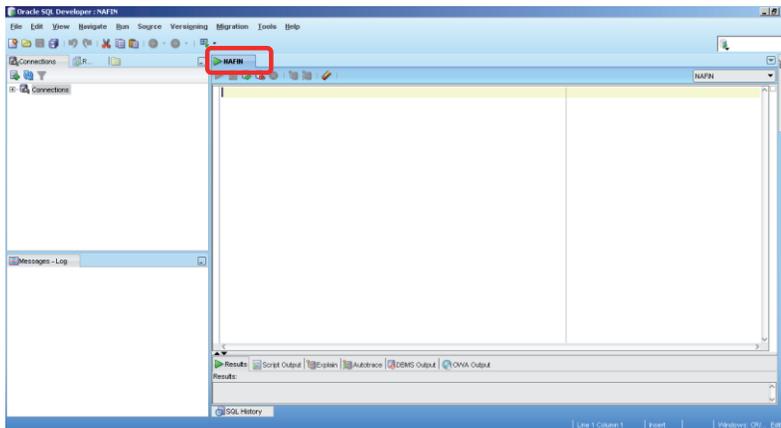


Fig. 3.13 Interfaz de desarrollo del *Developer*

3.7 Asignación de módulos a cada programador

En este punto se asignaron los módulos a migrar correspondientes a cada programador quedando de la siguiente manera destinados.

- Administración de recursos humanos
- Planeación
- Administración del pago
- Herramientas

3.7.1 Proyectos, reportes y stored procedure de administración de recursos humanos, planeación, administración del pago y herramientas

Para el módulo de administración del pago se contemplaron un total de:

- 130 proyectos a migrar
- 130 reportes
- Y un promedio de un stored procedure por reporte, dejando un estimado de 130 stored procedure también.

Para el módulo de herramientas se contemplaron un total de:

- 13 proyectos
- 6 reportes
- Y un promedio de un stored procedure por reporte, dejando un estimado de 6 stored procedure también.

Para el módulo de administración de recursos humanos se contemplaron un total del:

- 147 proyectos
- 438 reportes
- Y un promedio de un stored procedure por reporte, dejando un estimado de 438 stored procedure también.

Para el módulo de planeación se contemplaron un total del:

- 122 proyectos
- 163 reportes
- Y un promedio de un stored procedure por reporte, dejando un estimado de 163 stored procedure también.

3.8 Metodología de trabajo

La metodología que se ordenó a seguir fue la siguiente:

- 1) Se asignaba pantalla por pantalla de acuerdo al orden en que se encontraban en el módulo correspondiente, comenzando por los catálogos (Pantallas que generalmente contienen solo consultas) los cuales en estricta teoría son los más fáciles de migrar.
- 2) Se registraba en una bitácora el nombre y clave de la pantalla así como el nombre del programador y una fecha estimada de terminación dependiendo el grado de complejidad de la pantalla.
- 3) Se comenzaban a migrar las funciones o métodos que contenían instrucciones de SQL SERVER en la pantalla de *Visual Basic*.
- 4) Se migraban las clases relacionadas con la pantalla y que contenían instrucciones de SQL SERVER.
- 5) Se verificaba el número de reportes que cada pantalla contenía y a su vez se consultaba los stored procedure de cada reporte.
- 6) Se buscaba en SQL SERVER el o los *stored procedures* anteriormente detectados en los reportes encontrados. Una vez encontrados los stored procedures se analizaba que el nombre de los mismos no rebasara de 30 caracteres ya que en Oracle no se permite un nombre de procedimiento con longitud mayor de 30 caracteres. Si este rebasaba de lo permitido, se dejaba el nombre de tal manera que solo cumpliera con los 30 caracteres, pero tomando en cuenta que realmente se entendiera el nombre del stored procedure.

Nombre del stored procedure en SQL SERVER:

SP_EMPLEADOS_DATOS_COMPLEMENTARIOS

Nombre del stored procedure tentativo en Oracle:

SP_EMP_DATOS_COMPLEMENTARIOS

Fig 3.14 Stored procedure en SQL SERVER

En el ejemplo anterior se muestra como se abrevio el nombre del *stored procedure* de tal manera que se comprenda el nombre del mismo.

- 7) Se migraban los stored procedures

- 8) Se realizaban las pruebas únicamente verificando que no se marcara error en lo ya migrado.
- 9) Se armaba un zip con la pantalla, clases, *stored procedures* y un archivo de power point ilustrando las pruebas que se habían realizado a la pantalla en funcionamiento.
- 10) Se envía el zip a una carpeta contenedora.

3.9 Migración del módulo de administración del pago

En este punto se explicará de forma más detallada el proceso de migración que se llevó a cabo en todo el módulo.

Primero como ya se explicó anteriormente, se asignaba una pantalla dependiendo del grado de complejidad, se buscaban desde *Visual Basic* los métodos relacionados sobre una consulta, inserción, actualización, eliminación o en algunos casos un nombre de un proceso.

Ejemplo:

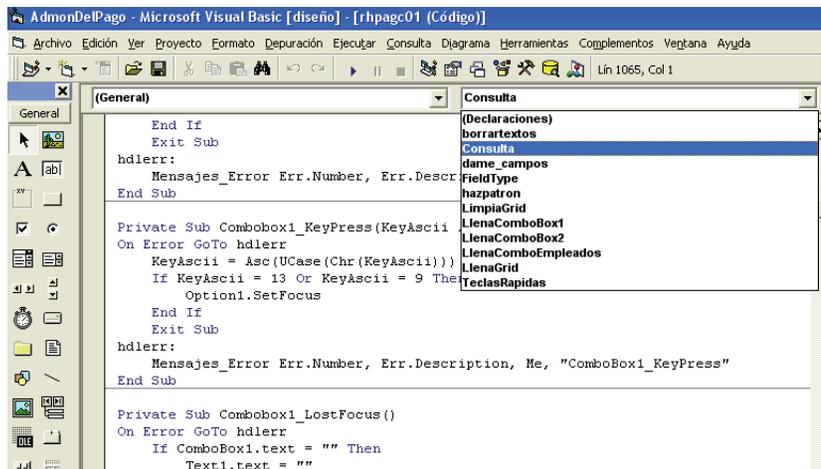


Fig. 3.15 Código de consulta en *Visual Basic* 6.0

Una vez detectado el método o función se procedía a migrar.

En la pantalla “Catalogo de conceptos” en la función “Consulta”, se consultan todos los conceptos existentes en la tabla correspondiente con opción a mandar un filtro por clave de concepto.

La forma como se arma la consulta en *Visual Basic* es dentro de una variable de tipo carácter (*String*). En la figura 3.16 Declaración de variables se muestra la manera en la que se hace en *Visual Basic 6.0*

```
Dim StrConsulta As String
Dim Contexto AsObjectContext
Dim Conectar As ADODB.Connection
Dim Rs As ADODB.Recordset
Dim StrConexion As String
```

Fig. 3.16 Declaración de variables

Dónde:

StrConsulta: Es el nombre de la variable la cual contiene la consulta.

Contexto: Es el objeto el cual indica si se realizó correctamente o incorrectamente la transacción.

Conectar: Objeto para la realización de la conexión.

Rs: Objeto en el cual se regresan los resultados de la consulta.

StrConexion: Variable donde se almacena la cadena de conexión hacia la base de datos.

A continuación se muestra en la figura 3.17 Configuración de consulta, la configuración y ejecución de una consulta de acuerdo a como se trabajó para este proyecto:

```
Dim StrConsulta As String
Dim Contexto As ObjectContext
Dim Conectar As ADODB.Connection
Dim Rs As ADODB.Recordset
Dim StrConexion As String
Dim GridConceptos as MSHFlexGrid
Set Contexto = GetObjectContext() → Asignación del contexto transaccional.
Set conectar = New ADODB.Connection → Asignación de propiedad tipo conexión.
StrConexion = "DSN=NAFIN;UID=SQLUSU2001;Pwd=k_23h45t"
Conectar.ConnectionString = StrConexion → Asignación de la cadena de conexión.
```

Conectar.CommandTimeout = 0 → Nos indica el tiempo de ejecución de la transacción. En este caso 0 equivale a tiempo ilimitado.

Conectar.Open --> Abre la conexión.

StrConsulta = "Select * From conceptos Where cve_concepto like '%'" --> Se asigna la cadena que contiene la consulta.

Set Rs = New Recordset --> Se asigna la propiedad del objeto tipo recordset.

rs.CursorLocation = adUseClient

Rs.Open StrConsulta, Conectar, adOpenDynamic, adLockReadOnly → Aquí se ejecuta la consulta y se regresan los resultados.

Contexto.SetComplete --> Completamos la transacción.

Set3 Conectar = nothing → Se libera el objeto de tipo conexión.

Set GridConceptos = Rs --> Se le asigna al grid el Recordset.

Set Rs = nothing → Se libera el objeto de tipo Recordset.

Fig. 3.17 Configuración de consulta

A continuación en la figura 3.18 Consulta SQL SERVER – Oracle, se presentan las instrucciones SQL que se migraron de la pantalla “Catalogo de conceptos”.

Ejemplo de consulta (Select con la cláusula Where):

Sintaxis SQL Server sin filtro:

```
Select * From conceptos Where cve_concepto like '%'
```

Sintaxis Oracle sin filtro:

```
Select * From conceptos Where cve_concepto like '%';
```

Sintaxis SQL Server con filtro:

```
Select * From conceptos Where cve_concepto like 'CONCEPTO1'
```

Sintaxis Oracle con filtro:

```
Select * From conceptos Where cve_concepto like 'CONCEPTO1';
```

Fig. 3.18 Consulta SQL SERVER- Oracle con where

En en la siguiente figura 3.19 Consulta con conversión de fecha, se consultan las claves de los conceptos seguido de un espacio y su descripción y la fecha de creación de dicho concepto en formato “dd MMM yyyy”:

Ejemplo de la consulta (Select con conversión de fechas)

Sintaxis SQL Server:

```
Select cve_concepto + ' ' + descripción, cast(varchar(11), fecha_creacion, 106)
From conceptos
```

Sintaxis Oracle:

```
Select cve_concepto || ' ' || descripción, to_char(fecha_creacion, 'dd MON yyyy')
From conceptos;
```

Fig. 3.19 Consulta con conversión de fecha

A continuación en la figura 3.20 Subconsulta en SQL SERVER – Oracle, se muestra una consulta que muestre el concepto 201, descripción y que muestre el nombre del empleado el cual inicio sesión en el sistema.

Consulta (Select, con una subconsulta y la cláusula where)

Sintaxis SQL Server:

```
Select cve_concepto, descripción, (select nombre + ' ' + apellido_paterno + ' ' +
apellido_materno From empleados Where registrado = 'S')
From conceptos
Where cve_concepto = '201'
```

Sintaxis Oracle:

```
Select cve_concepto, descripcion, (Select nombre || ' ' || apellido_paterno || ' ' ||
apellido_materno From empleados Where registrado = 'S')
From conceptos
Where cve_concepto = '201';
```

Fig. 3.20 Subconsulta en SQL SERVER - Oracle

En la figura 3.21 Consulta con ordenamiento y conversión, se muestra la clave de concepto, descripción de todos los conceptos ordenados por la clave del concepto.

Consulta (Select con un ordenamiento y una conversión numérica de la clave del concepto)

Sintaxis SQL Server:

```
Select cve_concepto, descripcion
From conceptos
```

```
Order by cast(cve_concepto as numeric)
Sintaxis Oracle.
Select cve_concepto, descripcion
From conceptos
Order by to_number(cve_concepto);
```

Fig. 3.21 Consulta con ordenamiento y conversión

Después de migrar todas las funciones de *Visual Basic* con instrucciones de SQL SERVER, se procede a migrar el o los stored procedures correspondientes a los reportes o procesos que la pantalla contiene. Para este caso la pantalla "Catalogo de conceptos", tiene un reporte donde se muestran todos los conceptos y su descripción que están dados de alta. A continuación se muestra la sintaxis básica en SQL SERVER y su equivalente en Oracle para el stored procedure llamado "SPConceptos":

En la figura 3.22 Stored procedure (SQL SERVER - Oracle), se da sintaxis general de *STORED PROCEDURE*.

```
Sintaxis SQL Server:
ALTER PROCEDURE [dbo].SPConceptos @ClaveConcepto
AS
BEGIN
Select * From Conceptos Where cve_concepto like @ClaveConcepto
END
Sintaxis Oracle:
CREATE OR REPLACE PROCEDURE " SPConceptos "
(
ClaveConcepto IN VARCHAR2 DEFAULT NULL,
Resultados IN OUT Owmb_emulation.globalpkg.RCT1
)
AS
BEGIN
OPEN RC1 FOR
Select * From conceptos Where cve_concepto like ClaveConcepto
END;
```

Fig. 3.22 Stored procedure (SQL SERVER - Oracle)

En la figura 3.23 Mandar llamar un *stored procedure* desde *Visual basic 6.0*, se muestra el bloque de código en *Visual Basic* para mandar llamar el SP migrado.

```
crystalreport1.ReportFileName = "\\NombreDelServidor\Lobo_RH\Administracion del  
pago\Reportes\NombreDelReporte.rpt" → le asigna la dirección completa donde se  
encuentra el reporte guardado en el servidor.
```

```
If TextoCveConcepto.text = "" then
```

```
    crystalreport1.StoredProcParam(0) = "%"
```

```
Else
```

```
    Crystalreport1.StoredProcParam(0) = TextoCveConcepto.text
```

```
End if → Si el texto llamado "TextoCveConcepto" no tiene valor, se entiende que no se  
ingresó ningún concepto, por lo tanto toma el valor "%" que se asigna a un arreglo de  
parámetros propio del objeto crystalreport1.
```

```
Crystalreport1.Action = 1 → Abre el reporte con extensión "rpt".
```

Fig. 3.23 Mandar llamar un *stored procedure* desde *Visual basic 6.0*

En otro de los catálogos llamado "Catalogo de fórmulas", se guardan los nombres de las formulas y una descripción. Estas fórmulas son procedimientos almacenados utilizados para calcular el sueldo, percepciones, deducciones y percepciones. Cada que se crea un concepto el cual se va a incorporar al cálculo de la nómina, se crea un *stored procedure* los cuales se ejecutan en la pantalla "Calculo de la nómina". A continuación se observan las instrucciones migradas de este catálogo.

En la figura 3.24 *Getdate()*, se muestra una consulta donde indica la clave de la formula, su descripción y su nombre de *stored procedure* relacionado a dicha fórmula y la fecha actual con opción a filtrar por clave de formula.

Consulta con ejemplo *GETDATE()*

Sintaxis SQL Server sin filtro:

```
Select cve_formula, descripcion, nombre_sp, getdate() From catalogo_formulas  
Where cve_formula like '%'
```

Sintaxis Oracle sin filtro:

```
Select cve_formula, descripcion, nombre_sp, sysdate From catalogo_formulas  
Where cve_formula like '%';
```

Sintaxis SQL Server con filtro:

```
Select cve_formula, descripcion, nombre_sp, getdate() From catalogo_formulas  
Where cve_formula like 'SP201'
```

Sintaxis Oracle con filtro:

```
Select cve_formula, descripcion, nombre_sp, sysdate From catalogo_formulas  
Where cve_formula like 'SP201';
```

Fig. 3.24 *Getdate()* en una consulta

El catalogo permite insertar una nueva fórmula con cajas de textos desde visual Basic que permiten introducir la clave de la formula, la descripción y el nombre del stored procedure. A continuación se muestra la figura 3.25 *Insert* (SQL SERVER - Oracle)

Sintaxis SQL Server:

```
Insert into catalogo_formulas values('SP023','Subsidio de gasolina','SP023')
```

Sintaxis Oracle:

```
Insert into catalogo_formulas values('SP023','Subsidio de gasolina','SP023');  
Commit;
```

Fig. 3.25 *Insert* (SQL SERVER - Oracle)

Permite también la actualización de su descripción de la formula y el nombre del *stored procedure*, tal como se muestra en la figura 3.26 *Update* a una formula, por si se llega a cambiar, solo de la clave de la fórmula que se seleccione o se indique.

Sintaxis SQL Server:

```
Update catalogo_formulas set descripcion = 'Subsidio de gasolina 2010', nombre_sp =  
'SP023' Where cve_fromula = 'SP023'
```

Sintaxis Oracle:

```
Update catalogo_formulas set descripcion = 'Subsidio de gasolina 2010', nombre_sp =  
'SP023' Where cve_formula = 'SP023';  
Commit;
```

Fig. 3.26 *Update* a una fórmula

Para eliminar una formula existente en el catálogo, tal como se muestra en la fórmula 3.27 *Delete* a una formula.

Sintaxis en SQL Server:

```
Delete From catalogo_formulas Where cve_formula = 'SP023'
```

Sintaxis en Oracle:

```
Delete From catalogo_formulas Where cve_formula = 'SP023';  
Commit;
```

Fig. 3.27 *Delete* a una formula

La pantalla tiene una opción tipo CheckBox la cual da la opción de borrar el *stored procedure* relacionado con la formula en el catálogo, tal como se muestra en la figura 3.28 *Drop* a un *stored procedure*.

Sintaxis en SQL Server:

```
Drop procedure [dbo].SP023
```

Sintaxis en Oracle:

```
Drop procedure "SP023";  
Commit;
```

Fig. 3.28 *Drop* a un *stored procedure*

En la siguiente pantalla, llamada "Asistencias" se consultan la clave de los empleados, su nombre completo el día de asistencia con formato "dd MMM yyyy" y el día de la semana, filtrado por la clave del empleado y el día.

En la figura 3.29 DATEPART en una consulta, se muestra cómo utilizar esta función con relación a las fechas que se manejan:

Sintaxis en SQL Server sin filtro:

```
Select a.cve_empleado, e. nombre + ' ' + e.apellido_paterno + ' ' +  
isnull(e.apellido_materno, ' '), cast(varchar(11), a.fecha, 106) as fecha, datepart(dw,  
getdate()) as dia_semana From asistencias a, empleados e Where e.cve_empleados =  
a.cve_empleado and a.cve_empleado like '%' and fecha = '2010-11-13 00:00:00'
```

Sintaxis en Oracle sin filtro:

```
Select a.cve_empleado, e. nombre || ' ' || e.apellido_paterno || ' ' ||  
NVL(e.apellido_materno, ' '), to_char(a.fecha, 'dd MON yyyy') as fecha, datepart('dw',  
sysdate) as dia_semana From asistencias a, empleados e Where e.cve_empleados =  
a.cve_empleado and a.cve_empleado like '%' and fecha = to_date('2010-11-13 00:00:00');
```

Sintaxis en SQL Server con filtro:

```
Select a.cve_empleado, e. nombre + ' ' + e.apellido_paterno + ' ' +  
isnull(e.apellido_materno, ' '), cast(varchar(11), a.fecha, 106) as fecha, datepart(dw,  
getdate()) as dia_semana From asistencias a, empleados e Where e.cve_empleados =  
a.cve_empleado and a.cve_empleado like '002323' and fecha = '2010-11-13 00:00:00'
```

Sintaxis en Oracle con filtro:

```
Select a.cve_empleado, e. nombre || ' ' || e.apellido_paterno || ' ' ||  
NVL(e.apellido_materno, ' '), to_char(a.fecha, 'dd MON yyyy') as fecha, datepart('dw',  
sysdate) as dia_semana From asistencias a, empleados e Where e.cve_empleados =  
a.cve_empleado and a.cve_empleado like '002323' and fecha = to_date('2010-11-13  
00:00:00');
```

Fig. 3.29 DATEPART en una consulta

En el *stored procedure* de esta pantalla, se utiliza una variable a la cual se le asigna la consulta para después ejecutarla, tal como se muestra en la figura 3.30 *Stored procedure* con variables.

Ejemplo de *stored procedure* con variables.

Sintaxis SQL Server:

```
ALTER PROCEDURE [dbo].SPAsistencias @ClaveEmpleado as varchar(6), @Fecha
as varchar(10)
AS
Declare @Cadena as varchar(4000)
BEGIN
Set @Cadena = 'Select * From asistencias Where cve_empleado like ''' +
@ClaveConcepto + ''' and fecha = ''' + cast(@Fecha as datetime) + ''' '
Exec(@Cadena)
END
```

Sintaxis Oracle:

```
CREATE OR REPLACE PROCEDURE " SPConceptos "
(
ClaveEmpleado IN VARCHAR2 DEFAULT NULL,
Fecha IN VARCHAR2 DEFAULT NULL,
RC1 IN OUT Owmb_emulation.globalpkg.RCT1
)
AS
Cadena VARCHAR2(4000);
BEGIN
Cadena := 'Select * From asistencias Where cve_empleado like ''' || ClaveEmpleado || '''
and fecha = ''' || to_date(Fecha) || '''
';
OPEN RC1 FOR (Cadena);
END;
```

Fig. 3.30 *Stored procedure* con variables

En la siguiente pantalla, llamada “Cambio de sueldos” se consultan la clave de los empleados, su nombre completo, sueldo, fecha de cambio y una columna del tipo de empleado donde pone ejecutivo en caso de que el empleado tenga un sueldo mayor a diez mil pesos, en caso contrario pone empleado.

En la siguiente figura 3.31 Consulta con *case – when*, se muestra lo mencionado anteriormente

```

Sintaxis SQL Server
Select s.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' + apellido_materno,
s.sueldo, s.fecha_cambio, (case when s.sueldo>10000 then
    'Ejecutivo'
Else
    'Empleado'
End) as Tipo_Empleado
From empleados e, sueldos s
Where e.cve_empleado = s.cve_empleado

Sintaxis Oracle:

Select s.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' || apellido_materno,
s.sueldo, s.fecha_cambio, (case when s.sueldo = 10000 then
    'Ejecutivo'
Else
    'Empleado'
End) as Tipo_Empleado
From empleados e , sueldos s
Where e.cve_empleado = s.cve_empleado

```

Fig. 3.31 Consulta con *case - when*

En la siguiente figura 3.32 *stored procedure* con cláusula *if*, para conocer las diferencias

```

Sintaxis SQL Server:
ALTER PROCEDURE [dbo].SPCAMBIOS_SUELDOS @ClaveEmpleado as
varchar(6), @Sueldo as numeric
AS
Declare @cadena as varchar(4000)
BEGIN
IF @Sueldo > 10000
BEGIN
Set @Cadena = ' select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' +
e.apellido_materno, s.sueldo, s.fecha_cambio,
'Ejecutivo' as tipo_empleado '
END

```

```
Else
BEGIN
Set @Cadena = ' select e.cve_empleado, e.nombre + ' ' + e.apeliido_paterno + ' ' +
e.apellido_materno, s.sueldo, s.fecha_cambio ,
'Empleado' as tipo_empleado'
END
Exec(@Cadena)
END
Sintaxis Oracle:
CREATE OR REPLACE PROCEDURE "SPCAMBIOS_SUELDOS"
(
  ClaveEmpleado IN VARCHAR DEFAULT NULL,
  Sueldo IN NUMBER DEFAULT NULL,
  RC1 IN OUT Owmb_emulation.globalpkg.RCT1
)
AS
Cadena VARCHAR2(4000);
BEGIN
IF Sueldo > 10000 THEN
Cadena := 'select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' ||
e.apellido_materno, s.sueldo, s.fecha_cambio,
'Ejecutivo' as tipo_empleado ';
ELSE
Cadena := ' select e.cve_empleado, e.nombre || ' ' || e.apeliido_paterno || ' ' ||
e.apellido_materno, s.sueldo, s.fecha_cambio ,
'Empleado' as tipo_empleado';
END;
OPEN RC1 FOR (Cadena);
```

Fig. 3.32 *Stored procedure* con cláusula *if*

Cuando se hace un cambio de sueldo en esta pantalla, también se realiza una inserción en la tabla "historicos_empleados" donde se lleva el historial de los empleados en la empresa, en la figura 3.33 Inserción a partir de un *select*, se muestra como se hace.

Sintaxis en SQL Server:

```
Insert into historicos_empleado
  Select e.cve_empleado, select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' '
+ e.apellido_materno, 'CAMBIO_SUELDO' as motivo, getdate() as fecha_transaccion
```

Sintaxis en Oracle:

```
Insert into historicos_empleado
  Select e.cve_empleado, select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' '
|| e.apellido_materno, 'CAMBIO_SUELDO' as motivo, sysdate as fecha_transaccion
```

Fig. 3.33 Inserción a partir de un *select*

En la siguiente pantalla llamada “Consulta de días de trabajo en el periodo” solo se consulta un *stored procedure* el cual muestra la clave del empleado, el nombre completo y los días que el empleado trabajó en el periodo (esta información se toma de la tabla *asistencias*), lo anterior se resume en la figura 3.34 *Stored procedure con while y datediff*.

Sintaxis en SQL Server:

```
ALTER PROCEDURE [dbo].SPDIASTRABAJO @ClaveEmpleado as varchar(6),
@Fecha_inicio_periodo as varchar(10), @Fecha_fin_periodo as varchar(10)
AS
Declare @DiasPeriodo as numeric
Declare @DiasTrabajo as numeric
Declare @Empleado as varchar(6)
BEGIN
Set @DiasPeriodo = datediff('dd', @ Fecha_inicio_periodo, @Fecha_fin_periodo) + 1
Set @DiasTrabajo = 0
While @DiasPeriodo > 0
Begin
  Set @Empleado = (Select isnull(cve_empleado, ' ') from asistencias where cve_empleado
= ClaveEmpleado and fecha = cast(@ Fecha_inicio_periodo as datetime) )
  IF @Empleado <> ' '
  Begin
    @DiasTrabajo = @DiasTrabajo + 1
  End
End
```

```
Set @DiasPeriodo = @DiasPeriodo - 1
Set @ Fecha_inicio_periodo = cast(@ Fecha_inicio_periodo as datetime) + 1
End
Select      @ClaveEmpleado,      @Fecha_inicio_periodo      as      fecha_inicio,
cast(@Fecha_fin_periodo as datetime) as fecha_fin, @DiasTrabajo as dias_trabajo
END
```

Sintaxis en Oracle:

```
CREATE OR REPLACE PROCEDURE "SPDIASTRABAJO"
( ClaveEmpleado IN VARCHAR2 DEFAULT NULL,
  Fecha_inicio_periodo IN VARCHAR2 DEFAULT NULL,
  Fecha_fin_periodo IN VARCHAR2 DEFAULT NULL,
  RC1 Owmb_emulation.globalpkg.RCT1
)
AS
DiasPeriodo number;
DiasTrabajo number;
Empleado varchar2(6);
Fecha_inicio_periodo2 date;
BEGIN
  DiasPeriodo := datediff(dd, Fecha_inicio_periodo, Fecha_fin_periodo) + 1;
  DiasTrabajo := 0;
  Fecha_inicio_periodo2 = to_date(Fecha_fin_periodo);
While DiasPeriodo > 0 LOOP
  Select isnull(cve_empleado, ' ')
  Into Empleado
  from asistencias where cve_empleado = ClaveEmpleado and fecha =
to_date(Fecha_fin_periodo2) );
  IF Empleado <> ' ' THEN
    DiasTrabajo := DiasTrabajo + 1;
  END IF;
  DiasPeriodo := DiasPeriodo - 1;
  Fecha_inicio_periodo2 := to_date(Fecha_inicio_periodo) + 1;
LOOP
```

```
OPEN RC1 FOR
Select      ClaveEmpleado,      Fecha_inicio_periodo2      as      fecha_inicio,
to_date(Fecha_fin_periodo) as fecha_fin, DiasTrabajo as dias_trabajo
END;
```

Fig. 3.34 *Stored procedure con while y datediff*

Una de las pantallas más importantes se llama “Descarga de la nómina” la cual realiza el proceso del cálculo del sueldo a los empleados. El proceso consiste en que se crea en cada stored procedure relacionado a la fórmula del concepto, una tabla temporal con nombre igual al nombre de la fórmula donde se almacena el nombre del empleado y su total dependiendo del concepto.

Para el concepto 201 “sueldo”, en el sp llamado SP201 se crea una tabla temporal llamada “SP201” con la clave del empleado y el total que es el resultado del proceso para calcular el sueldo. Al final del proceso se inserta en una tabla llamada “detalles_de_pago” todos los importes calculados en los *stored procedure*. A continuación se muestra un ejemplo de creación de tabla temporal.

En la figura 3.35 Creación de una tabla a partir de un *select*, se resume lo anteriormente mencionado.

Sintaxis en SQL Server:

```
Select p.cve_empleado, (p.sueldo * 15) as total, 1 as incidencias, 0 as exento
Into [dbo].SP201
From plazas p, empleados e
where p.cve_empleado = e.cve_empleado and e.baja = 'N' and p.status = 'A' and
p.cve_plaza = (Select max(p1.cve_plaza) from plazas p1 Where p.cve_empleado =
p1.cve_empleado)
```

Sintaxis en Oracle:

```
Create table SP201 as
Select p.cve_empleado, (p.sueldo * 15) as total, 1 as incidencias, 0 as exento
From plazas p, empleados e
```

```
Where p.cve_empleado = e.cve_empleado and e.baja = 'N' and p.status = 'A' and  
p.cve_plaza = (Select max(p1.cve_plaza) from plazas p1 Where p.cve_empleado =  
p1.cve_empleado);
```

Fig. 3.35 Creación de una tabla a partir de un *select*

En la siguiente figura 3.36 Inserción de una variable, se muestra la asignación de una consulta a una variable.

Ejemplo de inserción de una variable

Sintaxis en SQL Server:

```
Declare @Sueldo as numeric(13,2)  
Set @Sueldo = (Select sueldo From plazas Where cve_empleado = '002323' and status =  
'A')
```

Sintaxis en Oracle:

```
Sueldo number(13, 2)  
Select sueldo  
Into vSueldo  
From plazas Where cve_empleado = '002323' and status = 'A';
```

Fig. 3.36 Inserción de una variable

En la siguiente pantalla, llamada “Recibos de pago” se consultan la clave de los empleados, su nombre completo, conceptos considerados de percepción, conceptos considerados de deducción y la fecha de impresión del recibo con el mes completo y en español ya que en el sistema está configurado el formato de fechas en inglés para lo cual se tiene una función genérica que se encarga de convertir el mes en español.

En la siguiente figura 3.37 Consulta donde se llama a una función se ejemplifica lo que anteriormente se menciona

Sintaxis en SQL Server:

```
Select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' + e.apellido_materno,  
dp.cve_concepto, (select c.descripcion from conceptos c where dp.cve_concepto =
```

```

c.cve_concepto) as Desc_Concepto, (Case When dp.tipo = 'P' Then 'Percepción' When
dp.tipo = 'D' Then 'Deducción' Else 'Provisión' End ) as Tipo_Concepto, (DATEPART ('dd',
GETDATE()) + ' / ' + [dbo].FuncionNombreEsp (DATEPART ('mm', GETDATE())) + ' / ' +
DATEPART ('year', GETDATE()) ) AS Fecha_pago
From empleados e, detalles_de_pago dp, periodos_nomina pn
Where e.cve_empleado = dp.cve_empleado
And dp.cve_periodo = pn.cve_periodo
And pn.normal_extraordinario = 'N'

```

Sintaxis en Oracle:

```

Select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' || e.apellido_materno,
dp.cve_concepto, (select c.descripcion from conceptos c where dp.cve_concepto =
c.cve_concepto) as Desc_Concepto, (Case When dp.tipo = 'P' Then 'Percepción' When
dp.tipo = 'D' Then 'Deducción' Else 'Provisión' End ) as Tipo_Concepto, (DATEPART (dd,
SYSDATE) || ' / ' || FuncionNombreEsp (DATEPART (mm, SYSDATE)) || ' / ' || DATEPART
(yyyy, SYSDATE)) AS Fecha_pago
From empleados e, detalles_de_pago dp, periodos_nomina pn
Where e.cve_empleado = dp.cve_empleado
And dp.cve_periodo = pn.cve_periodo
And pn.normal_extraordinario = 'N';

```

Fig. 3.37 Consulta donde se llama a una función

A continuación se muestra la figura 3.38 Función que regresa los meses en español

Sintaxis SQL Server:

```

CREATE FUNCTION [dbo]. FuncionNombreEsp (
@Mes as numeric)
RETURNS Varchar(20)
AS
BEGIN
IF @Mes = 1
    RETURN 'Enero'
IF @Mes = 2

```

```
RETURN 'Febrero'  
IF @Mes = 3  
    RETURN 'Marzo'  
IF @Mes = 4  
    RETURN 'Abril'  
IF @Mes = 5  
    RETURN 'Mayo'  
IF @Mes = 6  
    RETURN 'Junio'  
IF @Mes = 7  
    RETURN 'Julio'  
IF @Mes = 8  
    RETURN 'Agosto'  
IF @Mes = 9  
    RETURN 'Septiembre'  
IF @Mes = 10  
    RETURN 'Octubre'  
IF @Mes = 11  
    RETURN 'Noviembre'  
IF @Mes = 12  
    RETURN 'Diciembre'
```

END

Sintaxis en Oracle:

```
CREATE OR REPLACE FUNCTION FuncionNombreEsp (  
Mes IN NUMBER )  
RETURN Varchar2(20)  
AS  
BEGIN  
IF Mes = 1 THEN  
    RETURN 'Enero'  
END IF;  
IF Mes = 2 THEN  
    RETURN 'Febrero'  
END IF;
```

```
IF Mes = 3 THEN
    RETURN 'Marzo'
END IF;
IF Mes = 4 THEN
    RETURN 'Abril'
END IF;
IF Mes = 5 THEN
    RETURN 'Mayo'
END IF;
IF Mes = 6 THEN
    RETURN 'Junio'
END IF;
IF Mes = 7 THEN
    RETURN 'Julio'
END IF;
IF Mes = 8 THEN
    RETURN 'Agosto'
END IF;
IF Mes = 9 THEN
    RETURN 'Septiembre'
END IF;
IF Mes = 10 THEN
    RETURN 'Octubre'
END IF;
IF Mes = 11 THEN
    RETURN 'Noviembre'
END IF;
IF Mes = 12 THEN
    RETURN 'Diciembre'
END IF;
END;
```

Fig. 3.38 Función que regresa los meses en español

Existe una pantalla llamada “Reportes de nómina” en la cual el empleado puede visualizar los reportes de sus pagos de nómina de todo el año. La información mostrada al empleado es consultada de muchas tablas, y que son consultas que no va a cambiar mucho en relación con otro empleado. Para esto se tiene una vista que su trabajo es contener columnas de muchas tablas y trabajar como si ese conjunto de columnas fueran una tabla. En esta pantalla se muestra la clave del empleado, nombre, su sueldo, clave de la empresa, nombre de la empresa, clave del área, nombre del área, clave del puesto, nombre del puesto, periodo de pago, fecha_inicio_nomina, fecha_fin_nomina y la fecha del mismo.

En la siguiente figura 3.39 vista dentro de un *select*, se muestra como se ejecuta esta como una consulta.

Sintaxis en SQL Server:

```
Select * From VistaReciboPagoEmpleadoPorPeriodo Where cve_empleado = '002323' and fecha_inicio_nomina = cast('2010-11-01' as datetime) and fecha_fin_nomina = cast('2010-11-15' as datetime)
```

Sintaxis en ORACLE:

```
Select * From VistaReciboPagoEmpPorPeriodo Where cve_empleado = '002323' and fecha_inicio_nomina = to_date('01 NOV 2010') and fecha_fin_nomina = to_date('15 NOV 2010');
```

Fig. 3.39 vista dentro de un *select*

En la siguiente figura 3.40 Creación del objeto tipo vista el cual se llama figura VistaReciboPagoEmpleadoPorPeriodo, se muestra como crear una vista tanto en SQL SERVER como Oracle.

Sintaxis en SQL Server:

```
CREATE VIEW [dbo].VistaReciboPagoEmpleadoPorPeriodo AS Select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' + e.apellido_materno as nombre, p.sueldo as sueldo_total, p.cve_empresa, (Select em.descripcion from empresas em Where em.cve_empresa = p.cve_empresa) as nombre_empresa, p.cve_area, (Select a.descripcion from catalogo_areas where p.cve_area = a.cve_area) as nombre_area, p.cve_puesto, (Select pu.descripcion from puestos pu Where pu.cve_puesto = p.cve_puesto) as
```

```
nombre_puesto, dp.cve_periodo as periodo_pago, dp.fecha_inicio_nomina,
dp.fecha_fin_nomina, dp.fecha_pago
From empleado e, plazas p, detalles_de_pago dp
Where e.cve_empleado = p.cve_empleado and p.cve_empleado = dp.cve_empleado

Sintaxis en Oracle:

CREATE OR REPLACE FORCE VIEW VistaReciboPagoEmpPorPeriodo (cve_empleado,
nombre, sueldo_total, cve_empresa, nombre_empresa, cve_area, nombre_area, cve_puesto,
nombre_puesto, periodo_pago, fecha_inicio_nomina, fecha_fin_nomina, fecha_pago)
AS
Select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' || e.apellido_materno as
nombre, p.sueldo as sueldo_total, p.cve_empresa, (Select em.descripcion from empresas em
Where em.cve_empresa = p.cve_empresa) as nombre_empresa, p.cve_area, (Select
a.descripcion from catalogo_areas where p.cve_area = a.cve_area) as nombre_area,
p.cve_puesto, (Select pu.descripcion from puestos pu Where pu.cve_puesto = p.cve_puesto)
as nombre_puesto, dp.cve_periodo as periodo_pago, dp.fecha_inicio_nomina,
dp.fecha_fin_nomina, dp.fecha_pago
From empleado e, plazas p, detalles_de_pago dp
Where e.cve_empleado = p.cve_empleado and p.cve_empleado = dp.cve_empleado;
```

Fig. 3.40 Creación de la vista VistaReciboPagoEmpleadoPorPeriodo.

Al igual que existe una pantalla donde se calcula la nómina a los empleado, también existe una pantalla llamada “Depuración de movimientos de nómina”. En dicha pantalla se le puede cargar a la nómina algún concepto que no se haya configurado desde el sistema, que sea nuevo o para agregar algún importe extra al empleado, etc. Al igual que se puede borrar algún concepto que se le haya calculado mal al empleado. A esta pantalla solo tiene acceso una persona asignada a esa función o algún administrador del sistema. A continuación se muestra el borrado de algún concepto tomando en cuenta más de una tabla. Ya que en Oracle no se puede realizar dicha acción como tal, para eso se realiza un ciclo donde se guarda el identificador de cada registro en un FOR y después se hace el delete de la tabla solo de los identificadores que el FOR contenga.

En la figura 3.41 Eliminando en dos tablas, se muestra cómo hacer un cursor para eliminar en dos tablas.

Sintaxis en SQL Server:

```
Delete dp
From detalles_de_pago dp, plazas p
Where dp.cve_empleado = p.cve_empleado And p.cve_empleado = '002323'
And p.cve_centro_proceso_nomina = '001' and dp.cve_periodo = '21'
And dp.cve_concepto = '201'
```

Sintaxis en Oracle:

```
BEGIN
FOR REC IN(Select dp.ROWID From detalles_de_pago dp, plazas p
Where dp.cve_empleado = p.cve_empleado And p.cve_empleado = '002323'
And p.cve_centro_proceso_nomina = '001' and dp.cve_periodo = '21'
And dp.cve_concepto = '201')
LOOP
Delete From detalles_de_pago dp Where dp.ROWID = REC.ROWID;
Commit;
END LOOP;
END;
```

Fig. 3.41 Eliminando en dos tablas

Existe una pantalla llamada “Cambio masivo de sueldos” se da la opción de que se le pueda actualizar el sueldo a todos los empleado del centro de proceso nomina 002 por da un ejemplo. Que en oracle no se puede hacer un update de más de 2 tablas como tal. El caso del aumento de sueldo del 20% a todos los empleados del centro de proceso nómina 001 y de la empresa Pantera Software.

En la figura 3.42 Actualizando en dos tablas, se muestra cómo hacer un cursor para realizar la actualización en las dos tablas.

Sintaxis en SQL Server:

```
Update p set p.sueldo = p.sueldo + (p.sueldo * 0.20)
From plazas p, empresas e Where p.cve_empresa = e.cve_empresa And
p.cve_centro_proceso_nomina = '001' and e.descripcion like 'PANTERA SOFTWARE'
```

Sintaxis en Oracle:

```
BEGIN
FOR REC IN(Select p.ROWID, (p.sueldo + (p.sueldo * 0.20)) as nuevo_sueldo
From plazas p, empresas e Where p.cve_empresa = e.cve_empresa
And p.cve_centro_proceso_nomina = '001'
And e.descripcion like 'PANTERA SOFTWARE')
LOOP
Update plazas p set p.sueldo = REC.nuevo_sueldo
Where p.ROWID = REC.ROWID;
Commit;
END LOOP;
END;
```

Fig. 3.42 Actualizando en dos tablas

Dentro de las diferencias entre SQL SERVER y Oracle en cuanto a su sintaxis, las que más se distinguen son los cursores. En la pantalla de programaciones de turnos, se puede insertar automáticamente a los empleados hasta fin de año actual la programación de su turno. Para esto se utiliza un *Stored procedure* dentro del proceso de inserción en el cual se emplea un cursor. En este caso se les quiere hacer una programación de turnos a los empleados activos del centro de proceso de nómina 001 que no tengan programación de turnos hasta antes del 05 de Julio del 2010 y que la programación de turnos se les realice hasta el último día del año actual.

En la figura 3.43 Ejecución de un cursor, se realiza esta acción donde se utilizan las clausulas *NOT IN*, *GROUP BY*

Sintaxis en SQL Server:

```
DECLARE @cve_empleado varchar(6)
DECLARE @CVE_TURNNO varchar(6)
DECLARE @fecha_ini datetime
DECLARE @Laborable varchar(6)
DECLARE CursorProgramacion CURSOR FOR
SELECT P.CVE_EMPLEADO, PT.CVE_TURNNO , MAX(PT.FECHA) fecha_ini
```

```
FROM PLAZAS p, PROGRAMACION_DE_TURNOS pt
WHERE p.cve_empleado = pt.cve_empleado
and p.STATUS = 'A'
AND PT.CVE_TURNO NOT IN('01', '02', '03')
AND p.CVE_EMPLEADO NOT IN(
    SELECT CVE_EMPLEADO
    FROM PROGRAMACION_DE_TURNOS
    WHERE FECHA >= '05 JUL 2010')
GROUP BY P.CVE_EMPLEADO, PT.CVE_TURNO
OPEN CursorProgramacion
FETCH NEXT FROM CursorProgramacion @cve_empleado, @CVE_TURNO, @fecha_ini
WHILE @@FETCH_STATUS = 0
BEGIN
    WHILE @fecha_ini <= '31 DEC 2010'
    BEGIN
        select @Laborable = laborable
        from turnos_dia
        where cve_turno = @CVE_TURNO
        and DIA = DATEPART(WEEKDAY, @fecha_ini)

        delete from PROGRAMACION_DE_TURNOS where cve_empleado =
@cve_empleado and fecha = @fecha_ini

        INSERT INTO PROGRAMACION_DE_TURNOS VALUES (@cve_empleado,
@fecha_ini, @CVE_TURNO, @Laborable, GETDATE())

        SET @fecha_ini = DATEADD(DAY, 1, @fecha_ini)
    END

    FETCH NEXT FROM CursorProgramacion into @cve_empleado, @CVE_TURNO,
@fecha_ini
END
CLOSE CursorProgramacion
DEALLOCATE CursorProgramacion
```

Sintaxis en Oracle:

```
Laborable varchar2(6);
Fecha_inicio date;
BEGIN
  FOR CursorProgramacion IN(SELECT P.CVE_EMPLEADO, PT.CVE_TURNO,
MAX(PT.FECHA) fecha_ini
  FROM PLAZAS p, PROGRAMACION_DE_TURNOS pt
  WHERE p.cve_empleado = pt.cve_empleado
  and p.STATUS = 'A'
  AND PT.CVE_TURNO NOT IN('01', '02', '03')
  AND p.CVE_EMPLEADO NOT IN(
    SELECT CVE_EMPLEADO
    FROM PROGRAMACION_DE_TURNOS
    WHERE FECHA >= '05 JUL 2010')
  GROUP BY P.CVE_EMPLEADO, PT.CVE_TURNO)
  LOOP
    Fecha_inicio = CursorProgramacion.fecha_ini;
    WHILE Fecha_inicio <= '31 DEC 2010' LOOP
  select laborable
  into Laborable
  from turnos_dia
  where cve_turno = CursorProgramacion.CVE_TURNO
  and DIA = DATEPART(WEEKDAY, Fecha_inicio);

  delete from PROGRAMACION_DE_TURNOS
  where cve_empleado = CursorProgramacion.CVE_EMPLEADO
  and fecha = Fecha_inicio;

  INSERT INTO PROGRAMACION_DE_TURNOS
  VALUES      (CursorProgramacion.CVE_EMPLEADO,      Fecha_inicio,
CursorProgramacion.CVE_TURNO, Laborable, SYSDATE);

  Commit;
```

```
Fecha_inicio := DATEADD('day', 1, Fecha_inicio);  
    END LOOP;  
END LOOP;  
END;
```

Fig. 3.43 Ejecución de un cursor

De esta manera se da por concluido el modulo correspondiente Administración del pago, de igual manera es importante mencionar que solo se tomaron como ejemplo los procesos de algunas pantallas, ya que en la mayoría de este módulo son procesos repetitivos.

Por otra parte se explica la sintaxis que se siguió desde *Visual Basic 6.0* para la ejecución de los ejemplos anteriormente mostrados (Consultas, *Stored Procedures*, Inserciones, Actualizaciones, Eliminación, etc.)

3.10 Migración del módulo de Herramientas

En el módulo de herramientas se tienen las pantallas que se utilizan para la administración del sistema como son los catálogos de usuarios que utilizan el sistema, los roles que tiene cada usuario, esto para limitar el acceso a ciertas pantallas como pueden ser el cálculo de nómina, depuración de movimientos de nómina, baja de empleados, etc. Estas pantallas de control utilizan por lo regular inserciones, consultas, modificación, eliminaciones con instrucciones relativamente sencillas. Las cuales son parecidas a los ejemplos ya mostrados anteriormente pero utilizando algunas cláusulas diferentes.

En esta pantalla llamada "Usuarios" se dan de alta los usuarios que ingresaran en el sistema, insertando una clave de usuario, una contraseña, la clave del empleado al cual pertenece su usuario, una pregunta y una respuesta para un eventual olvido de contraseña.

Anteriormente en el punto 5.9 "Migración del módulo de Administración del pago" se explicó la forma genérica como se configura y ejecuta una consulta desde *Visual Basic 6.0*. A continuación se mostrará cómo se ejecuta una inserción, actualización, borrado o alguna ejecución de un *Stored procedure* el cual contenga un conjunto de instrucciones para realizar algún proceso en específico.

En la figura 3.44 Ejecución de un *insert* desde *Visual Basic 6.0*, se demuestra cómo se hace este procedimiento.

```
Dim StrInserta As String
Dim Contexto AsObjectContext
Dim Conectar As ADODB.Connection
Dim StrConexion As String
Set Contexto = GetObjectContext()
Set Conectar = New ADODB.Connection
StrConexion = "DSN=NAFIN;UID=SQLUSU2001;Pwd=k_23h45t"
Conectar.ConnectionString = StrConexion
Conectar.CommandTimeout = 0
Conectar.Open
StrInserta = "Insert into usuarios
values('JPCUGI', pwdencrypt('Pa$$w0rd'), '002323', 'Deporte favorito', 'futbol')"
```

Fig. 3.44 Ejecución de un *insert* desde *Visual Basic 6.0*

Cuando se quiere saber la descripción de una tabla como las columnas, llaves primarias, llaves foráneas, etc. En SQL SERVER tanto en Oracle existe un comando con el cual se puede saber, dicho comando se muestra en la figura 3.45 Descripción de una tabla.

Sintaxis en SQL Server:

```
Sp_help Usuarios
```

Sintaxis en Oracle:

```
Desc Usuarios;
```

Fig. 3.45 Descripción de una tabla

En la pantalla "Búsqueda de Usuarios" se busca por clave de usuario los datos pertenecientes al usuario como sus datos relacionados al empleado ordenando por nivel de acceso el cual su campo en la tabla es de tipo carácter pero en este caso se convierte en tipo numérico y donde se observa un ejemplo de un *select* actuando como tabla.

En la figura 3.46 Select en el from se muestra como un select puede estar actuando como tabla.

Sintaxis en SQL Server:

```
Select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' + e.apellido_materno as nombre, T.clave_usuario, T.pregunta, respuesta, T.nivel_acceso
```

```
From empleados e, (Select cve_usuario as clave_usuario, cve_empleado as clave_empleado, pregunta, respuesta, nivel_acceso From usuarios Where cve_usuario = 'JPCUGI') T
```

```
Where e.cve_empleado = T.clave_empleado
```

Sintaxis en Oracle:

```
Select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' || e.apellido_materno as nombre, T.clave_usuario, T.pregunta, respuesta, T.nivel_acceso
```

```
From empleados e, (Select cve_usuario as clave_usuario, cve_empleado as clave_empleado, pregunta, respuesta, nivel_acceso From usuarios Where cve_usuario = 'JPCUGI') T
```

```
Where e.cve_empleado = T.clave_empleado;
```

Fig. 3.46 Select en el from

Cuando se crea una nueva pantalla, se registra para que pueda aparecer en el sistema. A continuación se muestra la forma como aparecen las pantallas en el sistema:

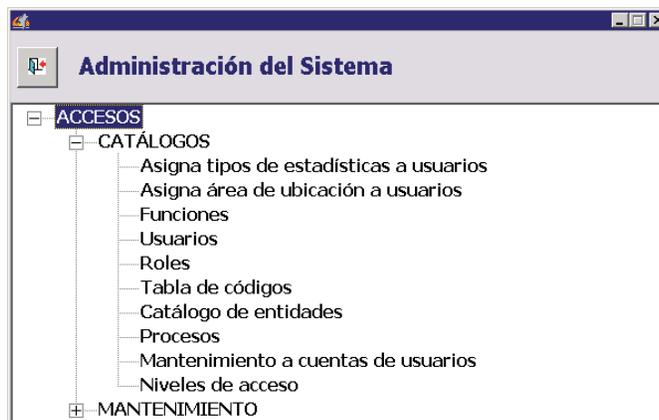


Fig. 3.47 Administración del sistema

Para esto se cuenta con la pantalla llamada Catalogo de funciones donde se guarda la información de la pantalla con su nombre y un número de posición en el menú como se muestra en la pantalla de arriba, para saber en qué número de pantalla es el máximo, se consulta el máximo número de la misma tabla. Esto se hace en un script (un archivo de texto con extensión *.sql) y se manda junto con el paquete como ya se había explicado anteriormente.

En la figura 3.48 Máxima posición de un menú, se escribe un script para sacar dicha posición utilizando la cláusula *MAX* y conversión a tipo numérico.

Sintaxis en SQL Server:

```
Declare @Maximo as numeric
Set @Maximo = (Select max(cast(menu_posicion as numeric))
From funciones
Where cve_menu = 'ADMSIS')

Insert into funciones('ADMSIS', 'RHADSF08', 'PANTALLA_NUEVA', @Maximo)
```

Sintaxis en Oracle:

```
Declare Maximo number;
BEGIN
Select max(to_number(menú_posicion))
Into Maximo
From funciones
Where cve_menu = 'ADMSIS';

Insert into funciones('ADMSIS', 'RHADSF08', 'PANTALLA_NUEVA', Maximo);

Commit;
END;
```

Fig. 3.48 Máxima posición de un menú

Teniendo la pantalla nueva registrada, se le asigna dicha pantalla a él o los usuarios los cuales van a trabajar con ella, en "Asignación de funciones a usuarios" se puede asignar

una función(Pantalla) a uno o más usuarios, lo que se hace es crear una tabla temporal con las claves de los usuarios y después insertar a la tabla correspondiente estos usuarios con la función seleccionada, como este proceso se repite muy seguido, se tiene para este caso una tabla de sesión, la cual solo contiene los datos en lo que dura el proceso.

En la figura 3.49 Tabla de sesión, se muestra como asignar funciones a usuarios

Sintaxis en SQL Server:

```
Create table #Usuarios (  
    cve_usuario varchar(6) not null  
)
```

Sintaxis en Oracle:

```
Create global temporary T_Usuarios(  
    cve_usuario varchar2(6) not null  
) on commit preserve rows;
```

Fig. 3.49 Tabla de sesión

La pantalla solo guarda la clave del usuario, la clave de la función y la fecha de asignación.

En la figura 3.50 Inserción a una función se realiza la misma a la tabla usuarios con la función seleccionada.

Sintaxis en SQL Server:

```
Insert into UsuariosFunciones  
Select cve_usuario, 'RHADSF08', getdate()  
From Usuarios  
Where cve_usuario IN(Select cve_usuario From #Usuarios)
```

Sintaxis en Oracle:

```
Insert into UsuariosFunciones  
Select cve_usuario, 'RHADSF08', sysdate  
From Usuarios  
Where cve_usuario IN(Select cve_usuario From T_Usuarios)
```

Fig. 3.50 Inserción a una función

Cuando se utiliza una tabla temporal creada desde algún stored procedure, como por ejemplo utilizar campos de varias columnas. Es necesario borrarla antes de ejecutar de nuevo el proceso. En la pantalla "Bases de datos", se consultan las bases de datos que los usuarios tienen asignadas ya que el sistema da la opción de acceder a diferentes bases de datos. A continuación se muestra un ejemplo de la cláusula *if exists*.

En la figura 3.51 *If exist* se muestra como se realiza un *if exist* en SQL SERVER y Oracle.

Sintaxis en SQL Server:

```
IF EXISTS(Select table_name
          From information_schema.tables
          Where table_name = 'UsuariosBasesDatos')
BEGIN
    Drop Table [dbo].UsuariosBasesDatos

    Select cve_usuario, 'BASEDATOS1', '192.0.0.50', getdate()
    Into [dbo].UsuariosBasesDatos
    From #Usuarios
END
```

Sintaxis en Oracle:

```
BEGIN
    IF EXISTS('Select object_name From user_objects
             Where object_name = "UsuariosBasesDatos " ') THEN
        EXEC('Drop table UsuariosBasesDatos');

        EXEC('Create table UsuariosBasesDatos as
             Select cve_usuario, 'BASEDATOS1', '192.0.0.50', sysdate
             From T_Usuarios');

    Commit;
```

```
END IF;  
END;
```

Fig. 3.51 *If exist*

Con esto se da por concluido el módulo de Herramientas, de igual manera es importante mencionar que solo se tomaron como ejemplo los procesos de algunas pantallas, ya que en la mayoría de este módulo son procesos repetitivos.

3.11 Migración del módulo de administración de recursos humanos

De igual manera, se sigue el mismo procedimiento que se explicó correspondiente a los módulos que anteriormente se migraron, es decir:

Se buscaban desde *Visual Basic* los métodos relacionados sobre una consulta, inserción, actualización, eliminación o en algunos casos un nombre de un proceso, una vez que se detectaba el método o función se procedía a migrarlo.

En la pantalla “Tipos de plaza” en la función “consulta”, se consultan todas las plazas existentes de la empresa determinada, en su tabla correspondiente con opción a mandar un filtro por clave de plaza.

En la siguiente figura 3.52 Consulta desde *Visual Basic 6.0*, se muestra como se realiza la misma desde el lenguaje de programación.

```
Dim StrConsulta As String  
Dim Contexto AsObjectContext  
Dim Conectar As ADODB.Connection  
Dim Rs As ADODB.Recordset  
Dim StrConexion As String
```

Fig. 3.52 Consulta desde *Visual Basic 6.0*

Dónde:

StrConsulta: Es el nombre de la variable la cual contiene la consulta.

Contexto: Es el objeto el cual indica si se realizó correctamente o incorrectamente la transacción.

Conectar: Objeto para la realización de la conexión.

Rs: Objeto en el cual se regresan los resultados de la consulta.

StrConexion: Variable donde se almacena la cadena de conexión hacia la base de datos.

De igual manera se explica cómo es que se realiza la consulta y se ejecuta la misma dentro del código de la pantalla con la finalidad de que quede más claro el proceso que se sigue en cada una de las pantallas migradas durante este y los demás módulos.

En la figura 3.53 Consulta y ejecución, se explica la manera en la que se arma la consulta y se ejecuta la misma.

Dim StrConsulta As String

Dim Contexto AsObjectContext

Dim Conectar As ADODB.Connection

Dim Rs As ADODB.Recordset

Dim StrConexion As String

Dim GridConceptos as MSHFlexGrid

Set Contexto = GetObjectContext() → Asignación del contexto transaccional.

Set conectar = New ADODB.Connection → Asignación de propiedad tipo conexión.

StrConexion = "DSN=NAFIN;UID=SQLUSU2001;Pwd=k_23h45t"

Conectar.ConnectionString = StrConexion → Asignación de la cadena de conexión.

Conectar.CommandTimeout = 0 → Nos indica el tiempo de ejecución de la transacción.

En este caso 0 equivale a tiempo ilimitado.

Conectar.Open --> Abre la conexión.

StrConsulta = "Select * From plazas Where cve_plaza like '%'" --> Se asigna la cadena que contiene la consulta.

Set Rs = New Recordset --> Se asigna la propiedad del objeto tipo recordset.

rs.CursorLocation = adUseClient

Rs.Open StrConsulta, Conectar, adOpenDynamic, adLockReadOnly → Aquí se ejecuta la consulta y se regresan los resultados.

```
Contexto.SetComplete --> Completamos la transacción.  
Set Conectar = nothing → Se libera el objeto de tipo conexión.  
Set GridConceptos = Rs --> Se le asigna al grid el Recordset.  
Set Rs = nothing → Se libera el objeto de tipo Recordset.
```

Fig. 3.53 Consulta y ejecución.

Como se mencionó, también se pueden realizar inserciones, actualizaciones, y eliminaciones dentro de las pantallas, es por eso que se considera importante también ilustrarlo y explicarlo.

En la próxima figura 3.54 Ejecución de *insert*, se muestra como se realiza esta desde *Visual Basic 6.0*.

```
Dim StrInserta As String  
Dim Contexto AsObjectContext  
Dim Conectar As ADODB.Connection  
Dim StrConexion As String  
  
Set Contexto = GetObjectContext()  
Set Conectar = New ADODB.Connection  
StrConexion = "DSN=NAFIN;UID=SQLUSU2001;Pwd=k_23h45t"  
Conectar.ConnectionString = StrConexion  
Conectar.CommandTimeout = 0  
Conectar.Open  
StrInserta = "Insert into usuarios  
values('040102', 'JESÚS RAFAEL ABUNDIS OCHOA', '24' 'JUAREZ 1001', '59620'  
'5122236', 'ING. SISTEMAS', 'DESARROLLADOR JR')"  
Conectar.Execute StrInserta  
Set Conectar = Nothing
```

Fig. 3.54 Ejecución de *insert*

Ahora se muestran las instrucciones SQL que se migraron en la pantalla "Tipos de plaza"

En la figura 3.55 Ejecución de consultas con *where*, se muestra la manera en la que se realizan consultas tanto en SQL SERVER como en Oracle.

Sintaxis SQL Server sin filtro:

```
Select * From plazas Where cve_plaza like '%'
```

Sintaxis Oracle sin filtro:

```
Select * From plazas Where cve_plaza like '%';
```

Sintaxis SQL Server con filtro:

```
Select * From plazas Where cve_plaza like 'PLAZA'
```

Sintaxis Oracle con filtro:

```
Select * From plazas Where cve_plaza like 'PLAZA' ;
```

Fig. 3.55 Ejecución de consultas con *where*

En la figura 3.56 Conversión de fechas, se muestra la sintaxis de cómo se realiza una conversión de fechas tanto en SQL SERVER como Oracle.

Sintaxis SQL Server:

```
Select cve_plaza + ' ' + descripción, cast(varchar(11), fecha_creacion, 106)  
From plazas
```

Sintaxis Oracle:

```
Select cve_plaza || ' ' || descripción, to_char(fecha_creacion, 'dd MON yyyy')  
From plazas;
```

Fig. 3.56 Conversión de fechas

En la siguiente consulta se mostrará la clave de plaza 113, descripción de la misma, así como el nombre del empleado que está asignado a esta plaza y que de igual manera se encuentre registrado.

En la próxima figura 3.57 Consultas y subconsultas, se muestra la sintaxis comparativa en ambos manejadores de base de datos.

Sintaxis SQL Server:

```
Select cve_plaza, descripción, (select nombre + ' ' + apellido_paterno + ' ' +
apellido_materno From empleados Where registrado = 'S')
From plazas
Where cve_plaza = '113'
```

Sintaxis Oracle:

```
Select cve_plaza, descripcion, (Select nombre || ' ' || apellido_paterno || ' ' ||
apellido_materno From empleados Where registrado = 'S')
From plazas
Where cve_plaza = '113';
```

Fig. 3.57 Consultas y subconsultas

En la consulta correspondiente a la figura 3.58 Consulta con ordenamiento, se muestra la clave de plaza, descripción de todas las plazas ordenados por la clave de la plaza.

```
Sintaxis SQL Server:
Select cve_plaza, descripcion
From plazas
Order by cast(cve_plaza as numeric)
```

Sintaxis Oracle.

```
Select cve_plaza, descripcion
From plazas
Order by to_number(cve_plaza);
```

Fig. 3.58 Consulta con ordenamiento.

Una vez finalizado la migración de la pantalla, se procede a migrar los stored procedure correspondientes a los reportes que se encontraron en dicha pantalla, en este caso por tratarse de un catálogo solo se encontró un reporte, por consiguiente un stored procedure donde realiza una consulta e indica las plazas y su descripción.

En la figura siguiente 5.59 Comparativo de *stored procedure*, se muestra el SP_Plazas, y sus diferentes sintaxis correspondientes.

Sintaxis SQL Server:

```
ALTER PROCEDURE [dbo]. SP_Plazas @CvePlaza
AS
BEGIN
Select * From Plazas Where cve_plaza like @CvePlaza
END
```

Sintaxis Oracle:

```
CREATE OR REPLACE PROCEDURE " SP_Plazas "
(
    CvePlaza IN VARCHAR2 DEFAULT NULL,
    RC1 IN OUT Owmb_emulation.globalpkg.RCT1)
AS
BEGIN
OPEN RC1 FOR
Select * From plazas Where cve_plaza like CvePlaza
END;
```

Fig. 3.59 Comparativa de *stored procedure*

De la figura 3.60 Ejecución del *stored procedure* desde *Visual Basic* 6.0, se da un ejemplo de cómo se hace este proceso.

crystalreport1.ReportFileName = "\\NombreDelServidor\Lobo_RH\Administracion de recursos humanos\Reportes\NombreDelReporte.rpt" → le asigna la dirección completa donde se encuentra el reporte guardado en el servidor.

If TextoCveConcepto.text = "" then

crystalreport1.StoredProcParam(0) = "%"

Else Crystalreport1.StoredProcParam(0) = TextoCveConcepto.text

End if → Si el texto llamado "TextoCveConcepto" no tiene valor, se entiende que no se ingresó ningún concepto, por lo tanto toma el valor "%" que se asigna a un arreglo de parámetros propio del objeto crystalreport1.

Crystalreport1.Action = 1 → Abre el reporte con extensión ".rpt".

Fig. 3.60 Ejecución del *stored procedure* desde *Visual Basic* 6.0

Por mencionar otro ejemplo, en el siguiente catálogo denominado “Catalogo de parentescos” se guardan todos los parentescos que un empleado tiene con respecto a su familia, como por ejemplo, esposa, hijos, padre, madre etc.

En la consulta correspondiente a la figura 3.61 Consulta con operador *like* se muestra la clave del parentesco, descripción, con la opción de poder filtrar la clave del parentesco.

```
Sintaxis SQL Server sin filtro:  
Select cve_parentesco, descripcion  
From parentescos Where cve_parentesco like '%'  
Sintaxis Oracle sin filtro:  
  
Select cve_parentesco, descripción  
From parentescos  
Where cve_parentesco like '%';  
Sintaxis SQL Server con filtro:  
  
Select cve_parentesco, descripción  
From catalogo_formulas  
Where cve_parentesco like 'HJO'  
Sintaxis Oracle con filtro:  
Select cve_parentesco, descripcion From parentesco Where cve_formula like 'HJO';
```

Fig. 3.61 Consulta con operador *like*

De igual manera es importante mencionar que en este catálogo se puede insertar parentescos, es decir de esta pantalla se dieron de alta todos los posibles parentescos que un empleado puede tener, a continuación se muestra la figura 3.62 *Insert* (SQL SERVER - Oracle).

```
Sintaxis SQL Server:  
Insert into parentescos values('HJO', 'HIJO')  
Sintaxis Oracle:  
Insert into catalogo_formulas values('ESPSA', 'ESPOSA');
```

```
Commit;
```

Fig. 3.62 Insert (SQL SERVER - Oracle)

Esta pantalla también permite actualizar los parentescos por si alguna vez se captura alguno de manera incorrecta en lugar de eliminarlo se puede actualizar, a continuación se muestra la figura 3.63 Actualización parentescos.

Sintaxis SQL Server:

```
Update parentescos set descripcion = 'ESPOSA'  
Where cve_parentesco = 'HPSA'
```

Sintaxis Oracle:

```
Update parentescos set descripcion = 'ESPOSA'  
Where cve_parentesco = 'HPSA'  
Commit;
```

Fig. 3.63 Actualización parentescos

La eliminación de algún parentesco se hace como se muestra en la figura 3.64 Eliminando parentescos:

Sintaxis en SQL Server:

```
Delete From parentescos Where cve_parentesco = 'HJA'
```

Sintaxis en Oracle:

```
Delete From parentescos Where cve_parentesco = 'HJA';  
Commit;
```

Fig. 3.64 Eliminando parentescos

En la siguiente pantalla, llamada "Alta de empleados" se consultan la clave de los empleados, su nombre completo el día que fueron registrados con formato "dd MMM yyyy" y el día de la semana, filtrado por la clave del empleado y el día, tal como se muestra en la figura 3.65 Consulta con *DATEPART*.

Sintaxis en SQL Server sin filtro:

```
Select ae.cve_empleado, e. nombre + ' ' + e.apellido_paterno + ' ' +  
isnull(e.apellido_materno, ' '), cast(varchar(11), ae.fecha_ingreso, 106) as fecha,  
datepart(dw, getdate()) as dia_semana  
From alta_empleados ae, empleados e  
Where e.cve_empleados = ae.cve_empleado and ae.cve_empleado like '%'  
and ae.fecha_ingreso = '2010-11-15 00:00:00'
```

Sintaxis en Oracle sin filtro:

```
Select ae.cve_empleado, e. nombre || ' ' || e.apellido_paterno || ' ' ||  
NVL(e.apellido_materno, ' '), to_char(ae.fecha_ingreso, 'dd MON yyyy') as fecha,  
datepart('dw', sysdate) as dia_semana  
From alta_empleados ae, empleados e  
Where e.cve_empleados = ae.cve_empleado  
and ae.cve_empleado like '%'  
and ae.fecha_ingreso = to_date('2010-11-15 00:00:00');
```

Sintaxis en SQL Server con filtro:

```
Select ae.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' +  
isnull(e.apellido_materno, ' '), cast(varchar(11), a.fecha, 106) as fecha, datepart(dw,  
getdate()) as dia_semana  
From alta_empleados ae, empleados e  
Where e.cve_empleados = ae.cve_empleado and a.cve_empleado like '002323' and  
ae.fecha_ingreso = '2010-11-15 00:00:00'
```

Sintaxis en Oracle con filtro:

```
Select ae.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' ||  
NVL(e.apellido_materno, ' '), to_char(a.fecha, 'dd MON yyyy') as fecha, datepart('dw',  
sysdate) as dia_semana  
From alta_empleados ae, empleados e  
Where e.cve_empleados = ae.cve_empleado and a.cve_empleado like '002323' and  
ae.fecha_ingreso = to_date('2010-11-15 00:00:00');
```

Fig. 3.65 Consulta con *DATEPART*

En la figura 3.66 *Stored procedure* se muestra el SPAIta_Empleados

Sintaxis SQL Server:2

```
ALTER PROCEDURE [dbo].SPAIta_Empleados
@ClaveEmpleado as varchar(6),
@Nombre as varchar(20),
@Sexo as varchar(2),
@FechaIngreso as varchar(10)
AS
Declare @Cadena as varchar(4000)
BEGIN
Set @Cadena = 'Select * From alta_empleados
Where cve_empleado like ''' + @ClaveEmpleado + '' , and sexo = ''' + @Sexo + ''''' and
fecha_ingreso = ''' + cast(@Fecha as datetime) + ''''
Exec(@Cadena)
END
```

Sintaxis Oracle:

```
CREATE OR REPLACE PROCEDURE " SPAIta_empleados "
(
    ClaveEmpleado IN VARCHAR2 DEFAULT NULL,
    Nombre IN VARCHAR2 DEFAULT NULL,
    Sexo IN VARCHAR2 DEFAULT NULL,
    FechaIngreso IN VARCHAR2 DEFAULT NULL,
    RC1 IN OUT Owmb_emulation.globalpkg.RCT1
)
AS
Cadena VARCHAR2(4000);
BEGIN
Cadena := 'Select * From alta_empleados Where cve_empleado like ''' || ClaveEmpleado
|| ''' and sexo = ''' || sexo || ''''' and fecha_ingreso = ''' || to_date(Fecha) || '''
';
OPEN RC1 FOR (Cadena);
END;
```

Fig. 3.66 *Stored procedure*

En la siguiente pantalla, llamada "Vacaciones_Empleados" se consultan el número de días que tiene un empleado de vacaciones, su clave de empleado, fecha_actual, antigüedad y una columna donde indica el número de días de vacaciones del empleado según su antigüedad, tal como se muestra en la figura 3.67 CASE - WHEN.

Sintaxis SQL Server

```
Select a.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' + e.apellido_materno,
ev.días, ev.fecha_actual, ev.antigüedad (case when ev.antigüedad.> 365 then
    'Ocho días otorgados'
When ev.antigüedad > 730
    'Doce días otorgados'
When ev.antigüedad >1095
    'catorce días otorgados'
Else
    'Aún no tienes vacaciones'
End) as vacaciones
From empleado e, empleados_vacaciones ev
Where e.cve_empleado = ev.cve_empleado
```

Sintaxis Oracle:

```
Select s.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' || e.apellido_materno,
ev.días, ev.fecha_actual, ev.antigüedad (case when ev.antigüedad = 365 then
    'Ocho días otorgados'
when ev.antigüedad = 730 then
    'Doce días otorgados'
when ev.antigüedad = 1095 then
    'catorce días otorgados'
Else
    'Aún no tienes vacaciones'
End) as vacaciones
```

```
From empleado e, empleados_vacaciones ev
Where e.cve_empleado = ev.cve_empleado
```

Fig. 3.67 CASE - WHEN

En la próxima figura 3.68 *Stored procedure* con *if* se muestra la comparativa en ambos manejadores

Sintaxis SQL Server:

```
ALTER PROCEDURE [dbo].SPVACACIONES @ClaveEmpleado as varchar(6),
@Antiguedad as numeric
AS
Declare @cadena as varchar(4000)
BEGIN
IF @Antiguedad > 365
BEGIN
Set @Cadena = ' select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' +
e.apellido_materno, ev.dias, ev_antiguedad,
' ocho días otorgados' as vacaciones '
END
IF @Antiguedad > 730
BEGIN
Set @Cadena = ' select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' +
e.apellido_materno, ev.dias, ev_antiguedad,
' doce días otorgados' as vacaciones '
END
IF @Antiguedad > 1095
BEGIN
Set @Cadena = ' select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' +
e.apellido_materno, ev.dias, ev_antiguedad,
' catorce días otorgados' as vacaciones '
END

Else
BEGIN
```

```
Set @Cadena = ' select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' +
e.apellido_materno, ev.dias, ev_antiguedad,
'sin vacaciones' as vacaciones '
END
Exec(@Cadena)
END
```

Sintaxis Oracle:

```
CREATE OR REPLACE PROCEDURE "SPCAMBIOS_SUELDOS"
```

```
(
```

```
    ClaveEmpleado IN VARCHAR DEFAULT NULL,
```

```
    Antiguedad IN NUMBER DEFAULT NULL,
```

```
    RC1 IN OUT Owmb_emulation.globalpkg.RCT1
```

```
)
```

```
AS
```

```
Cadena VARCHAR2(4000);
```

```
BEGIN
```

```
IF Antiguedad > 365 THEN
```

```
Cadena := 'select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' ||
e.apellido_materno, ev.dias, ev_antiguedad,
'ocho días de vacaciones' as vacaciones ;
```

```
IF Antiguedad > 730 THEN
```

```
Cadena := 'select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' ||
e.apellido_materno, ev.dias, ev_antiguedad,
'doce días de vacaciones as vacaciones ;
```

```
IF Antiguedad > 1095 THEN
```

```
Cadena := 'select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' ||
e.apellido_materno, ev.dias, ev_antiguedad,
'catorce días de vacaciones' as vacaciones ;
```

```
ELSE
```

```
Cadena := 'select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' ||  
e.apellido_materno, ev.dias, ev_antiguedad,  
'sin vacaciones ' as vacaciones ;  
END;  
OPEN RC1 FOR (Cadena);
```

Fig. 3.68 *Stored procedure con if*

Cuando se hace un cambio de sueldo en esta pantalla, también se realiza una inserción en la tabla “historicos_empleados” donde se lleva el historial de los empleados en la empresa, tal como se muestra en la figura 3.69 Inserción a partir de un select.

Sintaxis en SQL Server:

```
Insert into historicos_empleado  
Select e.cve_empleado, select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' '  
+ e.apellido_materno, 'CAMBIO_SUELDO' as motivo, getdate() as fecha_transaccion
```

Sintaxis en Oracle:

```
Insert into historicos_empleado  
Select e.cve_empleado, select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' '  
|| e.apellido_materno, 'CAMBIO_SUELDO' as motivo, sysdate as fecha_transaccion
```

Fig. 3.69 Inserción a partir de un select

Cuando a un empleado le corresponden vacaciones en esta pantalla, también se realiza una inserción en la tabla “historicos_empleados” donde se lleva el historial de empleados en la empresa.

En la siguiente pantalla llamada “Consulta de plazas en periodos” solo se consulta un stored procedure el cual muestra el clave del empleado, el nombre completo y la plaza que a tenido un empleado en un periodo determinado.

En la figura 3.70 *WHILE* y *DATEDIFF*, se muestra un *stored procedure* con el manejo de estas funciones.

Sintaxis en SQL Server:

```
ALTER PROCEDURE [dbo].SPDIATRABAJO @ClaveEmpleado as varchar(6),
@Fecha_inicio_periodo as varchar(10), @Fecha_fin_periodo as varchar(10), @Plaza as
varchar(20)
AS
Declare @Dias_Periodo as numeric
Declare @NPlazas as varchar(20)
Declare @Empleado as varchar(6)
BEGIN
Set @Dias_Periodo = datediff('dd', @ Fecha_inicio_periodo, @Fecha_fin_periodo) + 1

While @Dias_Periodo > 0
Begin
    Set @Empleado = (Select isnull(cve_empleado, ' ') from plazas_periodos pp, empleados
e where e.cve_empleado = @ClaveEmpleado and pp.fecha = cast(@Fecha_inicio_periodo
as datetime) and e.cve_empleado = pp.cve_empleado )

    Set @ Fecha_inicio_periodo = cast(@ Fecha_inicio_periodo as datetime) + 1
End
    Select      @ClaveEmpleado,      @Fecha_inicio_periodo      as      fecha_inicio,
cast(@Fecha_fin_periodo as datetime) as fecha_fin, @NPlazas as plazas_periodo
END
```

Sintaxis en Oracle:

```
CREATE OR REPLACE PROCEDURE "SPDIATRABAJO"
( ClaveEmpleado IN VARCHAR2 DEFAULT NULL,
  Fecha_inicio_periodo IN VARCHAR2 DEFAULT NULL,
  Fecha_fin_periodo IN VARCHAR2 DEFAULT NULL,
  Plaza IN VARCHAR2 DEFAULT NULL,
  RC1 Owmb_emulation.globalpkg.RCT1
)
AS
Dias_Periodo number;
NPlazas varchar2(20);
```

```
Empleado varchar2(6);
BEGIN
    DiasPeriodo := datediff(dd, Fecha_inicio_periodo, Fecha_fin_periodo) + 1;

While Dias_Periodo > 0 LOOP
    Select isnull(cve_empleado, ' ')
        Into Empleado
        from plazas_periodos pp, empleados e where e.cve_empleado = Clave_Empleado
and fecha = to_date(Fecha_fin_periodo) and e.cve_empleado = pp.cve_empleado );
LOOP
    OPEN RC1 FOR
        Select ClaveEmpleado, Fecha_inicio_periodo as fecha_inicio, to_date(Fecha_fin_periodo)
as fecha_fin, NPlazas as plazas_periodo
END;
```

Fig. 3.70 *WHILE* y *DATEDIFF*

Después de la pantalla “descarga de la nómina”, hay otra de igual importancia llamada “comisión por horas extras”, en la cual se da un bono económico a todos los empleados que trabajaron horas extras. Ya en el proceso a realizar existe un stored procedure para este cálculo relacionado a la fórmula correspondiente al concepto de horas extras, la creación de una tabla temporal con el mismo nombre que la fórmula, esto por estandar de desarrollo y el total del monto a recibir.

A continuación se explica el proceso a seguir para el cálculo de este bono:

El concepto "205" correspondiente a "horas extras", nos hará referencia para el nombre de nuestro stored procedure SP205 y se creará una tabla temporal denominada "SP205" por los motivos explicados anteriormente, esta tabla tendrá la clave del empleado, nombre, y el total que es el resultado del bono percibido.

En la siguiente figura 3.71 Tabla temporal, se muestra el comparativo tanto en SQL SERVER como en Oracle.

Sintaxis en SQL Server:

```
Select p.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' + e.apellido_materno,  
(Case When p.horas_extras = 'S' Then (p.sueldo * .20) Else (p.sueldo * .15) End ) as  
horas_extras  
Into [dbo].SP205  
From plazas p, empleados e  
where p.cve_empleado = e.cve_empleado and e.baja = 'N' and p.status = 'A' and  
p.cve_plaza = (Select max(p1.cve_plaza) from plazas p1 Where p.cve_empleado =  
p1.cve_empleado)  
  
Sintaxis en Oracle:  
Create table SP205 as  
Select p.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' || e.apellido_materno,  
(Case When p.horas_extras = 'S' Then (p.sueldo * .20) Else (p.sueldo * .15) End ) as  
horas_extras  
From plazas p, empleados e  
Where p.cve_empleado = e.cve_empleado and e.baja = 'N' and p.status = 'A' and  
p.cve_plaza = (Select max(p1.cve_plaza) from plazas p1 Where p.cve_empleado =  
p1.cve_empleado);
```

Fig. 3.71 Tabla temporal

En la siguiente figura 3.72 Inserción a una variable se muestra como se le asigna el valor de una consulta a una variable, en este caso a Total_Empleados.

Ejemplo de inserción a una variable.

Sintaxis en SQL Server:

```
Declare @Total_Empleados as numeric(8,2)
```

```
Set @Total_Empleados = (Select count(*) From empleados Where registrado = 'S')
```

Sintaxis en Oracle:

```
vSueldo number(8, 2)
```

```
Select count(*)
```

```
Into vTotal_Empleados
```

```
From empleados Where registrado = 'S';
```

Fig. 3.72 Inserción a una variable

En la siguiente pantalla "cursos de capacitación", se consulta la clave del empleado, nombre, cve_curso, nombre del curso, si el empleado es de planta o eventual y el tipo de capacitación recibida, es decir si fue por empresas asesores externos, dentro de la misma empresa, o en escuelas o centros de capacitación destinados a este fin, esto se explica en la figura 3.73.

Sintaxis en SQL Server:

```
Select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' + e.apellido_materno,
cc.cve_curso, (select te.descripcion from tipos_empleados where e.cve_empleado =
te.cve_empleado) as tipo_empleado, ([dbo].FuncionCursosCapa) AS Tipo_de_curso
From empleados e, cursos_capacitacion cc, catalogo_cursos c
Where e.cve_empleado = cc.cve_empleado
And cc.cve_curso = c.cve_curso
And e.registrado = 'S'
```

Sintaxis en Oracle:

```
Select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' || e.apellido_materno,
dp.cve_concepto, (select te.descripcion from tipos_empleados where e.cve_empleado =
te.cve_empleado) as tipo_empleado, ([dbo].FuncionCursosCapa) AS Tipo_de_curso
From empleados e, cursos_capacitacion cc, catalogo_cursos c
Where e.cve_empleado = cc.cve_empleado
And cc.cve_curso = c.cve_curso
And e.registrado = 'S';
```

Fig. 3.73 Función

Complementando la consulta anterior, en la figura 3.74 Función tipo curso, se mostrará la misma que hace referencia al tipo de capacitación.

Sintaxis SQL Server:

```
CREATE FUNCTION [dbo]. FuncionTipoCurso (
```

```
@TipoCurso as varchar(5))
RETURNS Varchar(20)
AS
BEGIN
IF @TipoCurso = 'EAE'
    RETURN 'EMPRESAS ASESORAS EXTERNAS'
IF @TipoCurso = 'DE'
    RETURN 'DENTRO DE LA EMPRESA'
IF @TipoCurso = 'ECC'
    RETURN 'ESCUELA O CENTRO DE CAPACITACIÓN'
END
```

Sintaxis en Oracle:

```
CREATE OR REPLACE FUNCTION FuncionTipoCurso (
TipoCurso IN VARCHAR2 DEFAULT NULL)
RETURN Varchar2(20)
AS
BEGIN
IF TipoCurso = 'EAE' THEN
    RETURN 'EMPRESAS ASESORAS EXTERNAS'
END IF;
IF TipoCurso = 'DE' THEN
    RETURN 'DENTRO DE LA EMPRESA'
END IF;
IF TipoCurso = 'ECC' THEN
    RETURN 'ESCUELA O CENTRO DE CAPACITACIÓN'
END IF;
END;
```

Fig. 3.74 Función tipo curso

En la pantalla “histórico de cursos” se puede consultar los cursos que ha tenido un empleado durante su trayectoria laboral, como esta información se contiene en varias tablas fue necesario tener una vista, ya que de esta manera se puede contener los datos de los

empleados, sus cursos y la fecha de inicio y de terminación del mismo. En la tabla se muestra la clave del empleado, su nombre, la clave del curso, nombre del curso fecha inicio y fecha fin del curso, tal como se muestra en la figura 3.75.

Sintaxis en SQL Server:

```
Select * From VistaHistoricoDeCursoPorEmpleado Where cve_empleado = '00022' and
fecha_inicio_curso = cast('2008-01-01' as datetime) and fecha_fin_curso = cast('2008-01-06'
as datetime)
```

Sintaxis en ORACLE:

```
Select * From VistaHistoricoDeCursoPorEmpleado Where cve_empleado = '00022' and
fecha_inicio_curso = to_date('01 ENE 2010') and fecha_fin_curso = to_date('06 ENE 2010');
```

Fig. 3.75 Vista

En la figura 3.76 Desarrollo de una vista, se muestra el proceso de la misma.

Sintaxis en SQL Server:

```
CREATE VIEW [dbo].VistaHistoricoDeCursoPorEmpleado AS Select e.cve_empleado,
e.nombre + ' ' + e.apellido_paterno + ' ' + e.apellido_materno as nombre, c.cve_curso as
clave_curso, e.descripcion as nombre_curso, hc.fecha_inicio_curso, hc.fecha_fin_curso
(Select em.nombre from empresas em Where em.cve_empresa = p.cve_empresa) as
nombre_empresa
From empleados e, cursos c, historico_cursos hc, plazas p
Where e.cve_empleado = p.cve_empleado
and c.cve_curso = hc.cve_curso
```

Sintaxis en Oracle:

```
CREATE OR REPLACE FORCE VIEW VistaHistoricoDeCursoPorEmpleado
(cve_empleado, nombre, clave_curso, nombre_curso, fecha_inicio_curso, fecha_fin_curso,
nombre_empresa
AS
Select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' || e.apellido_materno as
nombre, c.cve_curso as clave_curso, e.descripcion as nombre_curso, hc.fecha_inicio_curso,
```

```
hc.fecha_fin_curso, (Select em.nombre from empresas em Where em.cve_empresa =
p.cve_empresa) as nombre_empresa
From empleados e, cursos c, historico_cursos hc, plazas p
Where e.cve_empleado = p.cve_empleado
and c.cve_curso = hc.cve_curso;
```

Fig. 3.76 Desarrollo de una vista

Otra pantalla importante dentro de la administración de recursos humanos es la de “plan de vacaciones”, donde se puede determinar los días que un empleado desea tomar de su periodo vacacional, ahora, se pueden dar en repetidas ocasiones que el usuario capture mal los días a tomar, por lo cual implica realizar dos movimientos, eliminar de la tabla de detalle_empleados y vacaciones_empleado, es por eso que a continuación se muestra un ejemplo tanto en SQL SERVER como de Oracle de cómo realizar una eliminación de dos tablas, tal como se muestra en la figura 3.77 Eliminación multitable.

Sintaxis en SQL Server:

```
Delete de
From detalle_empleados de, vacaciones_empleado ve
Where de.periodo_vacacional = v.periodo_vacacional
And de.cve_empleado = ve.cve_empleado
And ve.status = 'A'
And ve.cve_concepto = '501'
```

Sintaxis en Oracle:

```
BEGIN
FOR REC IN(Select de.ROWID
           From detalle_empleados de, vacaciones_empleado ve
           Where de.periodo_vacacional = v.periodo_vacacional
           And de.cve_empleado = ve.cve-empleado
           And ve.status = 'A'
           And ve.cve_concepto = '501')
LOOP
Delete From detalle_empleados de Where dp.ROWID = REC.ROWID;
Commit;
```

```
END LOOP;  
END;
```

Fig. 3.77 Eliminación multitable

En la pantalla de “incapacidades” como su nombre lo indica se captura el periodo de incapacidad hacia un empleado cuando este lo requiera, para esto se debe impactar la ausencia de dicho empleado en dos tablas, “asistencias” e “incapacidades”, ya que de no hacerlo en la tabla de asistencias se le estaría registrando las faltas a los días que el empleado estuviera incapacitado, de la misma manera en la que se explica en la figura 3.78.

Sintaxis en SQL Server:

```
Update a set a.status = 'INC'  
From asistencias a, incapacidades i  
Where a.status = i.status  
And i.descripcion = 'INCAPACIDAD'
```

Sintaxis en Oracle:

```
BEGIN  
FOR REC IN(Select a.ROWID, a.status = 'INC' as status_empleado  
From asistencias a, incapacidades i  
Where a.status = i.status  
And i.descripcion = 'INCAPACIDAD')  
LOOP  
Update asistencias a  
set a.status = REC.status_empleado  
Where a.ROWID = REC.ROWID;  
Commit;  
END LOOP;  
END;
```

Fig. 3.78 Actualización multitable

3.12 Migración del módulo de Planeación

El módulo de planeación dentro del sistema, sirve para que el usuario pueda registrar nuevos grupos, empresas, niveles de organización, clasificar los niveles de organización,

áreas de responsabilidad, áreas de ubicación, puestos, etc. Realmente para la migración de todos los módulos se siguieron los mismos pasos que anteriormente se mencionaron, es por ello que solo se mostrará lo más relevante de este módulo, de tal manera de que no sea tan repetitivos.

En la pantalla “grupos” el usuario tiene la posibilidad de agregar grupos de la empresa, pero también los puede eliminar y/o actualizar, tal como se muestra en la figura 3.79.

```
Dim StrElimina As String
Dim Contexto AsObjectContext
Dim Conectar As ADODB.Connection
Dim StrConexion As String
Set Contexto = GetObjectContext()
Set Conectar = New ADODB.Connection
StrConexion = "DSN=NAFIN;UID=SQLUSU2001;Pwd=k_23h45t"
Conectar.ConnectionString = StrConexion
Conectar.CommandTimeout = 0
Conectar.Open
StrElimina = "delete from grupos where cve_grupo = 'NAFIN'"
Conectar.Execute StrInserta
Set Conectar = Nothing
```

Fig. 3.79 *Delete desde Visual basic 6.0*

A continuación se muestra el ejemplo de cómo se realiza la inserción de esta misma pantalla, de la misma manera en la que se muestra en la figura 3.80.

```
Dim StrInserta As String
Dim Contexto AsObjectContext
Dim Conectar As ADODB.Connection
Dim StrConexion As String

Set Contexto = GetObjectContext()
Set Conectar = New ADODB.Connection
```

```
StrConexion = "DSN=NAFIN;UID=SQLUSU2001;Pwd=k_23h45t"
Conectar.ConnectionString = StrConexion
Conectar.CommandTimeout = 0
Conectar.Open
StrInserta = "Insert into grupos
Values('NAFIN', 'NACIONAL FINANCIERA', 2010-11-18)"
Conectar.Execute StrInserta
Set Conectar = Nothing
```

Fig. 3.80 *Insert* a la tabla grupos

En la figura 3.81 Concatenación de campos, se podrán ver las diferencias que existen entre SQL SERVER y Oracle con respecto a cómo se concatenan algunas columnas, cabe mencionar que esto ya se había ilustrado en los ejemplos anteriores pero no se había explicado, además de cómo crear una tabla temporal mediante una subconsultas pero dentro del from principal de la consulta.

Sintaxis SQL Server:

```
Select e.cve_empleado, e.nombre + ' ' + e.apellido_paterno + ' ' + e.apellido_materno,
e.edad, e.direccion, tmp1.cve_grupo, tmp1.descripcion as nombre_grupo,
tmp2.cve_empresa, tmp2.descripcion as nombre_empresa
From empleados, (select cve_grupo, descripcion from grupos) as tmp1, (select
cve_empresa, descripción from empresas) as tmp2
Where e.cve_empleado = tmp1.cve_empleado
And e.cve_empleado = tmp2.cve_empleado
And e.registrado = 'S'
```

Sintaxis Oracle:

```
Select e.cve_empleado, e.nombre || ' ' || e.apellido_paterno || ' ' || e.apellido_materno,
e.edad, e.direccion, tmp1.cve_grupo, tmp1.descripcion as nombre_grupo,
tmp2.cve_empresa, tmp2.descripcion as nombre_empresa
From empleados, (select cve_grupo, descripcion from grupos) as tmp1, (select
cve_empresa, descripción from empresas) as tmp2
Where e.cve_empleado = tmp1.cve_empleado
```

```
And e.cve_empleado = tmp2.cve_empleado  
And e.registrado = 'S'
```

Fig. 3.81 Concatenación de campos

Como primer comentario es importante mencionar que en SQL SERVER la concatenación de columnas se representa por medio de los símbolos + ' ' + mientras que en Oracle se cambia por || ' ' || y las subconsultas dentro del from principal de una consulta permanecen de la misma manera.

Como se realiza una actualización de dos tablas (*UPDATE*) en SQL SERVER y en Oracle, en los otros módulos migrados se mostró como se hacía pero para que quede más entendido se muestra a continuación más gráficamente.

Suponiendo que se quiere actualizar el grupo de una empresa desde la pantalla que se llama “grupos/empresas” es importante que tanto en la tabla de grupos como en la tabla de empresas queden impactados estos cambios, ya que ambas manejan el grupo, es por eso que a continuación en la figura 3.82 *Update* (SQL SERVER - Oracle), se muestra como se hizo.

```
Sintaxis SQL Server  
Update g set g.cve_grupo = 'NAFINSOFT'  
From grupos g empresas em  
Where g.cve_grupo = em.cve_grupo  
And g.cve_grupo = 'NAFIN'  
And g.descripcion = 'NACIONAL FINANCIERA'
```

Fig. 3.82 *Update* (SQL SERVER - Oracle)

A continuación en la figura 3.83 *Update* de las tablas (grupos, empresas) se muestra el proceso de las mismas.

```
Sintaxis Oracle:  
BEGIN  
FOR REC IN (select g.ROWID, (g.cve_grupo = 'NAFINSOFT'))as nuevo_grupo  
From grupos g, empresas em
```

```
Where g.cve_grupo = e.cve_grupo
And g.cve_grupo = 'NAFIN'
And g.descripcion = 'NACIONAL FINANCIERA'

LOOP
  Update grupos g set g.cve_grupo = rec.nuevo_grupo
  Where g.ROWID = REC.ROWID;
Commit;
END LOOP
END;
```

Fig. 3.83 *Update* de las tablas (grupos, empresas)

Como se puede observar, para el caso de SQL SERVER no existe mayor problema para los updates de dos tablas, ya que se realizan de la misma manera que se hacen de una sola tabla, el problema se presenta cuando se hace en Oracle, ya que se tiene que usar un cursor para poder hacer el recorrido de la actualización.

Cuando se quiere saber la descripción completa de una tabla se emplea la siguiente instrucción como lo indica en la figura 3.84 Descripción de tablas.

Sintaxis en SQL Server:

```
Sp_help Usuarios
```

Sintaxis en Oracle:

```
Desc Usuarios
```

Fig. 3.84 Descripción de tablas

Estas instrucciones son muy utilizadas cuando se necesita saber todo sobre una tabla, como llaves primarias, llaves foraneas, columnas, etc.

Ahora se mostrará en la figura 3.85 Eliminando en las tablas (areas, areas_responsabilidad) como se realiza tanto en SQL SERVER como en Oracle, de igual manera se podrá observar que se hacen de manera diferente, para ello se tomará como ejemplo la pantalla de “clasificación de áreas de responsabilidad”, donde la tabla “áreas” esta

relacionada con la tabla `areas_responsabilidad`, cuando se da de baja un área de la tabla "áreas" es necesario también hacerlo en la tabla "areas_responsabilidad" ya que de lo contrario se generaría inconsistencia en la información.

Sintaxis en SQL Server:

```
Delete a
From areas a, areas_responsabilidad ar
Where a.cve_area = ar.cve_area
And a.cve_area = 'DPTODESA'
And a.descripcion = 'DEPARTAMENTO DE DESARROLLO'
```

Sintaxis en Oracle:

```
BEGIN
FOR REC IN(Select a.ROWID
From areas a, areas_responsabilidad ar
Where a.cve_area = ar.cve_area
And a.cve_area = 'DPTODESA'
And a.descripcion = 'DEPARTAMENTO DE DESARROLLO'
LOOP
Delete From areas a
Where a.ROWID = REC.ROWID;
Commit;
END LOOP;
END;
```

Fig. 3.85 Eliminando en las tablas (áreas, áreas_responsabilidad)

De igual manera que en los updates de dos tablas, en SQL SERVER no representa mayor problema hacerlo ya que se hace igual como cuando se elimina de una tabla, pero en Oracle se tiene que manejar nuevamente un cursor, como se mostró.

La manera en el que se hacen las conversiones en SQL SERVER y Oracle se maneja de manera diferente, ya sea para convertir una columna a tipo numérica o para hacerla hacia tipo carácter, es por eso que en el siguiente figura 3.86 Conversión de campos tipo

numérico, se indica el código con sus diferencias, para esto se tomará como ejemplo la pantalla de “plazas” al igual que la tabla que lleva el mismo nombre.

Sintaxis SQL Server:

```
select pl.cve_empleado
from plazas pl
where
    cast(pl.cve_plaza as numeric) = (
        select max(cast(pl1.cve_plaza as numeric))
        from plazas pl1
        where pl1.cve_empleado = pl.cve_empleado
    )
```

Sintaxis Oracle:

```
select pl.cve_empleado
from plazas pl
where
    to_number(pl.cve_plaza ) = (
        select max(to_number(pl1.cve_plaza))
        from plazas pl1
        where pl1.cve_empleado = pl.cve_empleado
    )
```

Fig. 3.86 Conversión de campos tipo numérico

En este pequeño ejemplo se mostró como se realiza una subconsulta dentro del where, y de igual manera se mostró como la conversión del campo para SQL Server se hace mediante un cast(NOMBRE_COLUMNNA AS TIPO_DE_DATO)), mientras que con Oracle se hace con un TO_NUMBER(NOMBRE_COLUMNNA).

Ahora en la figura 3.87 *Stored procedure* (SPGRUPOSEMPRESAS), se verá cual es la diferencia al crear un *stored procedure* en SQL SERVER y Oracle, para ello se tomará como ejemplo dos tablas “grupos” y “empresas” y se realiza una consulta sencilla donde se le pedirá la clave del grupo, el nombre del grupo, la clave de la empresa y el nombre de la empresa, filtrándole los parámetros que viene desde la pantalla que son, clave del grupo y clave de la empresa.

Sintaxis SQL Server

```
ALTER PROCEDURE [dbo].SPGRUPOSEMPRESAS
```

```
    @ClaveGrupo,
```

```
    @ClaveEmpresa
```

```
AS
```

```
BEGIN
```

```
    Select g.cve_grupo, g.descripcion as nombregrupo, e.cve_empresa,
```

```
    e.descripcion as nombreempresa
```

```
From grupos g, empresas e
```

```
Where g.cve_grupo = e.cve_grupo
```

```
And g.cve_empresa = e.cve_empresa
```

```
And g.cve_grupo like @ClaveGrupo
```

```
And e.cve_empresa like @ClaveEmpresa
```

Sintaxis Oracle:

```
CREATE OR REPLACE PROCEDURE "SPGRUPOSEMPRESAS"
```

```
{
```

```
    ClaveGrupo IN VARCHAR2 DEFAULT NULL,
```

```
    ClaveEmpresa IN VARCHAR2 DEFAULT NULL,
```

```
    RC1 IN OUT Owmb_emulation.globalpkg.RCT1
```

```
}
```

```
AS
```

```
BEGIN
```

```
OPEN RC1 FOR
```

```
Select g.cve_grupo, g.descripcion as nombregrupo, e.cve_empresa,          e.descripcion
```

```
as nombreempresa
```

```
From grupos g, empresas e
```

```
Where g.cve_grupo = e.cve_grupo
```

```
And g.cve_empresa = e.cve_empresa
```

```
And g.cve_grupo like ClaveGrupo
```

```
And e.cve_empresa like ClaveEmpresa
```

```
END;
```

Fig. 3.87 *Stored procedure (SPGRUPOSEMPRESAS)*

Con esto se da por concluido el módulo de planeación no sin antes mencionar que de este módulo todos los proceso de migración que se hacen son de manera igual, es decir, lo único que cambian son las pantallas y los proceso que cada una de ellas realiza, en si la estructura es la misma y como se mencionó anteriormente los pasos a migrar siempre son los mismos. Por otra parte es importante hacer mención que en estos módulos de migración se explicó lo más general en cuanto a que se tenía que migrar, teniendo bien claro y establecidos las sintaxis ya explicadas es suficiente para realizar la migración de todas las pantallas de los módulos y sus respectivos stored procedure y esta a su vez fue suficiente para migrar todo el sistema de recursos humanos.

3.13 Pruebas

Una vez terminados los módulos correspondientes a cada programador, se procedía a realizar las pruebas. Se tenían 2 bases de datos, una en SQL Server y la otra en Oracle donde se comparaban los datos de consultas, procesos, etc. Así como de cantidades arrojados al realizar el cálculo de la nómina de los empleados. Si los datos eran diferentes con los mismos parámetros en ambas tablas, se procedía a revisar nuevamente la pantalla a nivel solo de código Oracle.

Comentario [AISM2]: Quita la hoja en blanco!!! siguiente

CONCLUSIONES Y TRABAJO FUTURO

Conclusiones

Dentro del desarrollo de sistemas computacionales, es importante conocer diferentes lenguajes de programación de bases de datos donde se conocen los distintos comandos entre estos. En este proyecto se aprendió el lenguaje de bases de datos Oracle el cual en los últimos años se ha implementado con aplicaciones hechas en plataforma java. Para esta migración fue necesario conocer de forma básica gran parte de los comandos de SQL SERVER, así como comprender y entender los diagramas Entidad-Relación. Se tiene que aprender el lenguaje de bases de datos Oracle y saber adaptar las instrucciones de un lenguaje a otro o viceversa.

La práctica de realizar esta migración hace a uno capaz de resolver consultas, inserciones, *stored procedures*, funciones, etc. En SQL SERVER y Oracle diferenciando muy bien la sintaxis entre un lenguaje y otro. De la misma manera se aprende desde lenguaje de programación en este caso *Visual Basic 6.0* la diferencia de cómo se ejecuta una instrucción entre estos lenguajes. Se aprendió también lo que es desde la instalación de todas las herramientas base que fueron esenciales para el desarrollo, hasta realizar las pruebas con el cliente dando un periodo de tiempo para resolver cualquier observación que haya salido, sin duda fue un gran proyecto con el que se aprendieron conocimientos para lo que es la migración de lenguajes de bases de datos.

Por otra parte, se dejó bien en claro que fue lo que se tuvo que realizar para desarrollar este proyecto, que sin duda fue muy prolongado, no tanto por el grado de complejidad, sino por la magnitud del proyecto, espero que este proyecto sea de gran ayuda para futuras consultas.

BIBLIOGRAFÍA

- [1] Anónimo. *Aprenda visual Basic 6.0 como si estuviera en primero*. San Sebastias. 1999.
- [2] <http://usuarios.multimania.es/cursosgbd/UD4.htm> visitado en febrero y marzo de 2012.
- [3] <http://mit.ocw.universia.net/curso11208/11/11.208/IAP02/lecture-notes/lecture5-2.html> visitado en febrero y marzo de 2012.
- [4] <http://www.ajpsoft.com/modules.php?name=News&file=article&sid=223> visitado en febrero y marzo de 2012.
- [5] <http://www.oracleya.com.ar/index.php?inicio=0> visitado en febrero y marzo de 2012.
- [6] Cavero, J. M., Cuadra, D., Iglesias, A. M., & Nieto, C. *Bases de Datos Relacionales*. Pearson Prentice Hall. 2007.
- [7] Nicol, N., & Albrecht, R. *Todo Sobre Visual Basic 6.0*. Marcombo. 1999.
- [8] Perez Lopez, C. *Micorosft SQL Server 2005*. México: AlfaOmega. 2006.
- [9] Perez Lopez, C. *Oracle PL/SQL*. Rama, Librería y Editorial Mic. 2008.
- [10] RM, J. *Visual Basic 6.0*. 2007.
- [11] <http://javierm-visualbasic60.blogspot.com/2007/10/barra-de-herramientas.html> visitado febrero y marzo de 2012.
- [12] Roland Martínez, D., Valderas Aranda, P., & Martínez Gomez, E. *Oracle Básico*. Starbook. 2010.
- [13] Stanek, W. *SQL Server 2005 Manual del Administrador*. Mcgraw-hill. 2006.
- [14] Tschanz, D., Gunderloy, M., & Jorden, J. *SQL Server 2005*. Anaya. 2006.
- [15] Viera, R. *Fundamentos de Programación con SQL Server 2005*. Anaya Multimedia. 2006.

ÍNDICE DE FIGURAS

Fig. 1.0 Interpretación de un diagrama sintáctico	4
Fig. 1.1 Sheet of cheats SQL Server	5
Fig. 1.2 Sheet of cheats Visual Basic 6	6
Fig. 1.3 Sheet of cheats Oracle	10
Fig. 2.1 Enfoque top-down	14
Fig. 2.2 Diagrama entidad relación	15
Fig. 2.3 Instrucción SQL para crear una tabla	16
Fig. 2.4 Estándares de diagramación	19
Fig. 2.5 Matriz de relaciones	20
Fig. 3.1 Inicio instalación Oracle Developer	34
Fig. 3.2 Ubicación de archivos	34
Fig. 3.3 Instalación Oracle Developer	35
Fig. 3.4 Oracle services	35
Fig. 3.5 Ubicación de archivos	36
Fig. 3.6 Resumen	36
Fig. 3.7 Asistente de configuración	37
Fig. 3.8 Configuración ODBC	47
Fig. 3.9 Creación de nuevo origen de datos	47
Fig. 3.10 Driver	48
Fig. 3.11 Developer	49
Fig. 3.12 Configuración de la base de datos	49
Fig. 3.13 Interfaz de desarrollo del Developer	50
Fig. 3.14 Stored procedure en SQL SERVER	52
Fig. 3.15 Código de consulta en visual Basic 6.0	53
Fig. 3.16 Declaración de variables	54
Fig. 3.17 Configuración de consulta	55
Fig. 3.18 Consulta SQL SERVER- Oracle con where	55
Fig. 3.19 Consulta con conversión de fecha	56
Fig. 3.20 Subconsulta en SQL SERVER – Oracle	56
Fig. 3.21 Consulta con ordenamiento y conversión	57
Fig. 3.22 Stored procedure (SQL SERVER - Oracle)	58
Fig. 3.23 Mandar llamar un stored procedure desde visual basic 6.0	58
Fig. 3.24 Getdate() en una consulta	59
Fig. 3.25 Insert (SQL SERVER - Oracle)	59
Fig. 3.26 Update a una formula	60
Fig. 3.27 Delete a una formula	60
Fig. 3.28 Drop a un stored procedure	60
Fig. 3.29 DATEPART en una consulta	61
Fig. 3.30 Stored procedure con variables	62
Fig. 3.31 Consulta con case – when	63
Fig. 3.32 Stored procedure con clausal IF	65
Fig. 3.33 Inserción a partir de un select	65
Fig. 3.34 Stored procedure con while y datediff	67
Fig. 3.35 Creación de una tabla a partir de un select	68
Fig. 3.36 Inserción de una variable	68
Fig. 3.37 Consulta donde se llama a una función	69

Fig. 3.38 Función que regresa los meses en español	72
Fig. 3.39 Vista dentro de un select	72
Fig. 3.40 Creación de la vista VistaReciboPagoEmpleadoPorPeriodo	73
Fig. 3.41 Eliminando en dos tablas	74
Fig. 3.42 Actualizando en dos tablas	75
Fig. 3.43 Ejecución de un cursor	78
Fig. 3.44 Ejecución de un insert desde visual basic 6.0	79
Fig. 3.45 Descripción de una tabla	80
Fig. 3.46 Select en el from	80
Fig. 3.47 Administración del sistema	81
Fig. 3.48 Máxima posición de un menú	82
Fig. 3.49 Tabla de sesión	83
Fig. 3.50 Inserción a una función	83
Fig. 3.51 If exist	84
Fig. 3.52 Consulta desde visual basic 6.0	85
Fig. 3.53 Consulta y ejecución	86
Fig. 3.54 Ejecución de insert	87
Fig. 3.55 Ejecución de consultas con where	87
Fig. 3.56 Conversión de fechas	88
Fig. 3.57 Consultas y subconsultas	88
Fig. 3.58 Consulta con ordenamiento	89
Fig. 3.60 Ejecución del stored procedure desde visual basic 6.0	90
Fig. 3.61 Consulta con operador like	91
Fig. 3.62 Insert (SQL SERVER - Oracle)	91
Fig. 3.63 Actualización parentescos	91
Fig. 3.64 Eliminando parentescos	92
Fig. 3.65 Consulta con DATEPART	93
Fig. 3.66 Stored procedure	94
Fig. 3.67 CASE – WHEN	95
Fig. 3.68 Stored procedure con if	97
Fig. 3.69 Inserción a partir de un select	98
Fig. 3.70 WHILE y DATEDIFF	99
Fig. 3.71 Tabla temporal	101
Fig. 3.72 Inserción a una variable	101
Fig. 3.73 Función	102
Fig. 3.74 Función tipo curso	102
Fig. 3.75 Vista	103
Fig. 3.76 Desarrollo de una vista	104
Fig. 3.77 Eliminación multitable	105
Fig. 3.78 Actualización multitable	106
Fig. 3.79 delete desde visual basic 6.0	107
Fig. 3.80 Insert a la tabla grupos	107
Fig. 3.81 Concatenación de campos	108
Fig. 3.82 Update (SQL SERVER - Oracle)	109
Fig. 3.83 Update de las tablas (grupos, empresas)	109
Fig. 3.84 Descripción de tablas	110
Fig. 3.85 Eliminando en las tablas (áreas, áreas_responsabilidad)	111
Fig. 3.86 Conversión de campos tipo numérico	112
Fig. 3.87 Stored procedure (SPGRUPOSEMPRESAS)	118