PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

SEDE AMBATO

ESCUELA DE INGENIERIA DE SISTEMAS

DISERTACIÓN DE GRADO PREVIA LA OBTENCIÓN DEL TITULO DE INGENIERO DE SISTEMAS

"DB2 UDB V7.1 FAMILY FUNDAMENTALS "

EDWIN MARCELO GALLARDO PAREDES

EDGAR JAVIER VILLACRÉS VENEGAS

DIRECTOR DE LA DISERTACIÓN

ING. WIGBERTO SANCHEZ

Miriam Villeri de Mora

AMBATO, 2004

PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR
SEDE AMBATO
SECRETARIA
ESCUELA DE INGENIERIA
DE SISTEMAS

PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR
PROGRAMAS ACADEMICOS AMBATO
BIBLIOTECA

PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

SEDE AMBATO

ESCUELA DE INGENIERIA DE SISTEMAS

DISERTACIÓN DE GRADO PREVIA LA OBTENCIÓN DEL TITULO DE INGENIERO DE SISTEMAS

"DB2 UDB V7.1 FAMILY FUNDAMENTALS "

DIRECTOR: _____

Ing. Wigberto Sánchez

EDWIN MARCELO GALLARDO PAREDES

EDGAR JAVIER VILLACRÉS VENEGAS

AMBATO, 2004

DEDICATORIA

Esta tesis representa el trabajo constante de ser cada día mejor, es por ello que quiero dedicar primeramente a Dios por darme la fuerza y la sabiduría necesaria para seguir adelante.

A mis padres quienes me han enseñado a luchar para conseguir lo que se quiere, aunque lejos estaban siempre estuvieron pendientes de mi y que supieron transmitirme todo su Fe, comprensión pero sobre todo su amor, y que día a día fueron alimentando mis ilusiones para convertirlas en realidad, cuyo esfuerzo y sacrificio jamás podré retribuir lo suficiente en toda mi vida, y quiero decirles que siempre les llevo en mi mente y en mi corazón.

A toda mi familia quienes me han aconsejado y me han dado ánimo cuando más lo necesitaba y han hecho de mi una persona responsable y luchadora capaz de superar cualquier obstáculo que se presentaba en mi vida.

A todos mis maestros por darme un ejemplo de profesionalidad en especial a la Ing. Natasha Bayas por brindarme toda su paciencia y comprensión, así como también a mi director de tesis Ing.Ms. Wigberto Sánchez Peña que más que un profesor ejemplar es un gran amigo el cual admiro y respeto mucho.

Y a todas aquellas personas que de una u otra forma, colaboraron en la realización de este proyecto muchas gracias a todos.

Marcelo Gallardo Paredes

DEDICATORIA

El presente trabajo, fruto de mi esfuerzo y capacidad dedico primeramente a Dios quien me ha guiado y bendecido en mis estudios, por un camino de constante lucha, sacrificio y superación.

A mis padres que día a día me enseñaron a ser responsable y además me brindaron una oportunidad para surgir y avanzar.

A toda mi familia quienes me dieron el aliento necesario para la culminación de este trabajo.

A todos mis profesores en especial a la Ing. Natasha Bayas y el Ing. Wigberto Sánchez que desde el inicio de la disertación nos apoyaron y nos guiaron en todo momento.

Y como olvidarme de mis compañeros y amigos con los cuales hemos compartido momentos inolvidables en nuestra carrera universitaria.

Javier

AGRADECIMIENTO

Agradecemos a la Pontificia Universidad Católica del Ecuador Sede Ambato quien nos acogió en sus aulas, así como también a todos y cada uno de los distinguidos maestros quienes nos impartieron sus conocimientos dando un ejemplo de profesionalidad y deseos de superación como es el Ing. Patricio Medina quien nos ha demostrando ser un buen amigo y profesional.

A la Ing. Natasha Bayas quien desde el inicio de este proyecto nos supo guiar e incentivar con sus palabras de aliento para el avance de esta tesis.

Un agradecimiento muy especial aquellas personas que de manera desinteresada nos colaboraron con la información necesaria para el avance de este proyecto como son el Ing. Pablo Ruja Romero (Business Development Manager) de la empresa Nexsys del Ecuador, a la Ing. Silvana Villarroel ejecutiva de la IBM Quito, al Ing. Eduardo Gamboa jefe de Sistemas del Banco del Pichincha matriz Quito, Ing. Nadia Jaramillo que mas allá de ser una profesora ha sido una gran amiga, ganándose nuestro respeto y admiración.

A nuestro director de tesis Ing. Wigberto Sánchez Peña que gracias a sus conocimientos hemos llegado a la culminación de este trabajo.

Marcelo Gallardo Paredes *Javier Villacrés Venegas*

Índice

Página

Introducción 1

CAPÍTULO I

1. Proyecto de Estudio

1.1 Planteamiento del problema 3

1.2 Delimitación 3

1.2 Justificación 4

1.3 Objetivos 4

1.3.1 General 4

1.3.2 Específicos 5

1.4 Hipótesis 5

1.5 Metodologías de Investigación 5

CAPÍTULO II

2. Marco Teórico 6

2.1 DB2 Planning 6

2.1.1 DB2 Products 6

2.1.2 DB2 Tools 7

2.1.3 Control Center 9

2.1.4 The Configuration Assistent 10

2.1.5 Other DB2 Tools 11

2.2 DB2 Security 13

2.2.1 DB2 Authentication 13

2.2.2 Authorities 13

2.2.3 DB2 Privileges 14

2.3 Accessing DB2 UDB Data 14

2.3.1 What Makes up a DB2 Database 14

2.3.2 Creating your first Data Base 15

2.3.3 Manipulating DB2 Object with the Control Center 15

2.3.4 Creating and Accessing basic DB2 Object 16
2.4 Working with DB2 UDB Data 16
2.4.1 Structure Query Language (SQL) 16
2.4.2 Data Control Language (DCL) 16
2.4.3 Data Definition Language (DDL) 16
2.4.4 Data Manipulation Language (DML) 17
2.5 Working with DB2 UDB Object 17
2.5.1 Data Type 17
2.5.2 Tables 19
2.5.3 Constraint 19
2.5.4 Views 19
2.5.5 Indexes 19
2.6 Concurrency 20

CAPÍTULO III

3. Desarrollo del Proyecto

3.1 Manual DB2 UDB Family Fundamentals 21
3.2 Ejercicios 290

CAPÍTULO IV

4.1 Conclusiones 381
4.2 Recomendaciones 382
4.3 Validación 383
Bibliografía 389
Anexos 391

Índice de Figuras

	Página
DB2 Products	7
DB2 Tools (Show SQL)	8
DB2 Tools (Show Related)	9
Control Center	10
The Configuration Assistant	11
Other DB2 Tools	12
Manipulating DB2 Object UIT the Control Center	15
Data Type (Numeric)	17
Data Type (String)	18
Data Type (Date)	18

INTRODUCCION

Los sistemas de gestión de base de datos han sido bastante utilizados para organizar y manipular grandes cantidades de datos, estos sistemas son potentes programas para ordenador que pueden gestionar eficazmente un enorme número de elementos de datos.

Una base de datos es información útil organizada de una forma específica, por lo que se ha convertido en una herramienta que no puede faltar en las empresas, grande, mediana o pequeña.

Uno de los sistemas de gestión de base de datos reconocidos a nivel mundial es el DB2 UDB (Universal Data Base) que fue creado por la IBM (Internacional Business Machines Corporation).

DB2 es un sistema manejador de bases de datos relacionales (RDBMS) que ayuda a los usuarios a crear, actualizar y controlar bases de datos relacionales utilizando el Lenguaje Estructurado de Consultas (SQL).

DB2 fue también diseñado para reunir la información necesaria de pequeñas, medianas y grandes empresas sobre una gran variedad de plataformas entre las que se encuentran:

- Grandes sistemas como son: z/OS, OS/390, VM, y VSE.
- Sistemas medianos como: OS/400, AIX, Linux, HP-UX, y Solaris, y en sistemas pequeños: Windows 2000, Windows NT, Windows XP y Windows .NET.

Todas las versiones de la familia del DB2 tienen similares interfaces y utiliza similares algoritmos; es por esto que los clientes pueden utilizar el sistema en las diferentes plataformas.

Teniendo en cuenta todo lo antes mencionado y viendo las necesidades de nuestro medio nosotros hemos decidido en esta tesis elaborar un manual de DB2 el cual consiste en un compendio de diez capítulos, cada uno de los cuales se encuentran diseñados con diferentes imágenes de acuerdo al tema que se trate, para que de esta manera el estudiante pueda captar de una forma fácil, sencilla y entendible lo que va aprender.

Al final de cada capítulo hay un cuestionario para que el estudiante ponga en práctica lo aprendido. Así como también elaboramos un pequeño libro de trabajo el cual contiene ejercicios referentes al manual.

Todo este documento esta hecho en herramientas básicas como Word y PowerPoint para un buen entendimiento del mismo. Para garantizar la información que contiene el manual decidimos obtener la certificación internacional en dicha herramienta.

PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR
PROGRAMAS ACADEMICOS AMBATO
BIBLIOTECA

CAPITULO I

PROYECTO DE ESTUDIO

1.1 PLANTEAMIENTO DEL PROBLEMA

El problema se centra en el aprendizaje de la herramienta debido a su complejidad, en cuanto a su utilización, toda su documentación será netamente investigativa de parte del estudiante, en este caso compete realizar toda la investigación de los fundamentos o herramientas que se incluye en el DB2.

Al concluir la investigación se deberá estar en condiciones de poder manejar el paquete informático y así como la elaboración de un documento para que sirva de ayuda a las personas que estén interesadas en aprender esta herramienta informática.

Básicamente el área para donde está enfocada esta investigación comprende el estudio del módulo DB2 UDB V7.1 Family Fundamentals.

1.2 DELIMITACION

DB2 UDB V7.1 Family Fundamentals abarca varias especialidades como son:

- DB2 Planning
- DB2 Security
- Accessing DB2 UDB Data
- Working with DB2 UDB Data
- Working with DB2 UDB Object
- Data Concurrency

Debido al gran tamaño que el proyecto abarca el aprendizaje de los fundamentos de DB2 Universal Data Base versión 7.1.

1.3 JUSTIFICACION

Esta investigación servirá de ayuda al estudiante que tomará como una alternativa para su aprendizaje.

Dicho software requiere ser investigado y dominado para poder desarrollar las aplicaciones.

El estudio garantizará una nueva técnica de aprendizaje, la cual nos ayudará a una mejor preparación y dominio del software.

Día a día debemos ampliar ese círculo de atención para brindar los conocimientos que ésta nos ofrece y saber cómo y cuando utilizar, esa experiencia para el bien común para de esta manera permitir el desarrollo no solo del individuo, sino de la sociedad y por ende del país.

La certificación es costosa debido a que los cursos de capacitación son emitidos por la IBM.

1.4 OBJETIVOS

1.4.1 GENERAL:

- Investigar y aprender a utilizar la herramienta informática para posteriormente rendir un examen y obtener una certificación de la IBM en DB2 UDB V7.1 Family Fundamentals.

1.4.2 ESPECIFICOS:

- Explotar cada una de las características del DB2 UDB V7.1 para obtener provecho de ellas y aplicarlas posteriormente.
- Garantizar los conocimientos que se impartirán a los alumnos.
- Realizar un manual de usuario que sea fácil de manejar y que contenga las características del software para su posterior aprendizaje.
- Lograr que el estudio del software garantice el funcionamiento y utilización del mismo.
- Con el estudio de este software se garantiza una mejor preparación y dominio de sistemas en la vida profesional.
- Preparar a nuevos alumnos para que obtengan una certificación en este producto informático.

1.5 HIPOTESIS

El estudio de DB2 UDB V7.1 permitirá dar el examen en la IBM con el fin de obtener una certificación Internacional en esta herramienta y ser especialistas en la misma, para posteriormente capacitar a nuevos alumnos que estén interesados en los fundamentos de DB2.

1.6 METODOLOGIAS DE INVESTIGACION

Por el tipo de trabajo que se va a realizar, la metodología de investigación que se utilizará será el auto aprendizaje, por lo que nos convertiremos en autodidactas.

También se utilizará información extraída desde el Internet, bibliografía existente en el mercado y consultas a profesionales.

CAPITULO II

MARCO TEORICO

IBM es el mayor proveedor mundial de soluciones informáticas integrales de negocios. El software de DB2 se ejecuta en cualquier plataforma, desde una laptop hasta la supercomputadora más grande. Puede ser utilizada para administrar todo tipo de información, desde la de tipo personal hasta la de gigantescas corporaciones.

2.1 DB2 PLANNING

DB2 nos facilita la utilización de una amplia gama de productos existentes en el mercado, cada una de ellas nos brindan soluciones a nuestros problemas de acuerdo a sus necesidades.

2.1.1 DB2 PRODUCTS

DB2 versión 7.1 entrega soluciones de manejo de datos para cualquier negocio. Ningún otro sistema manejador de base de datos puede contener un rendimiento avanzado en cuanto habilidad, escalabilidad y manejabilidad características que DB2 V7.1 provee.

Existen diferentes versiones de DB2 disponibles en nuestro medio como muestra el grafico.

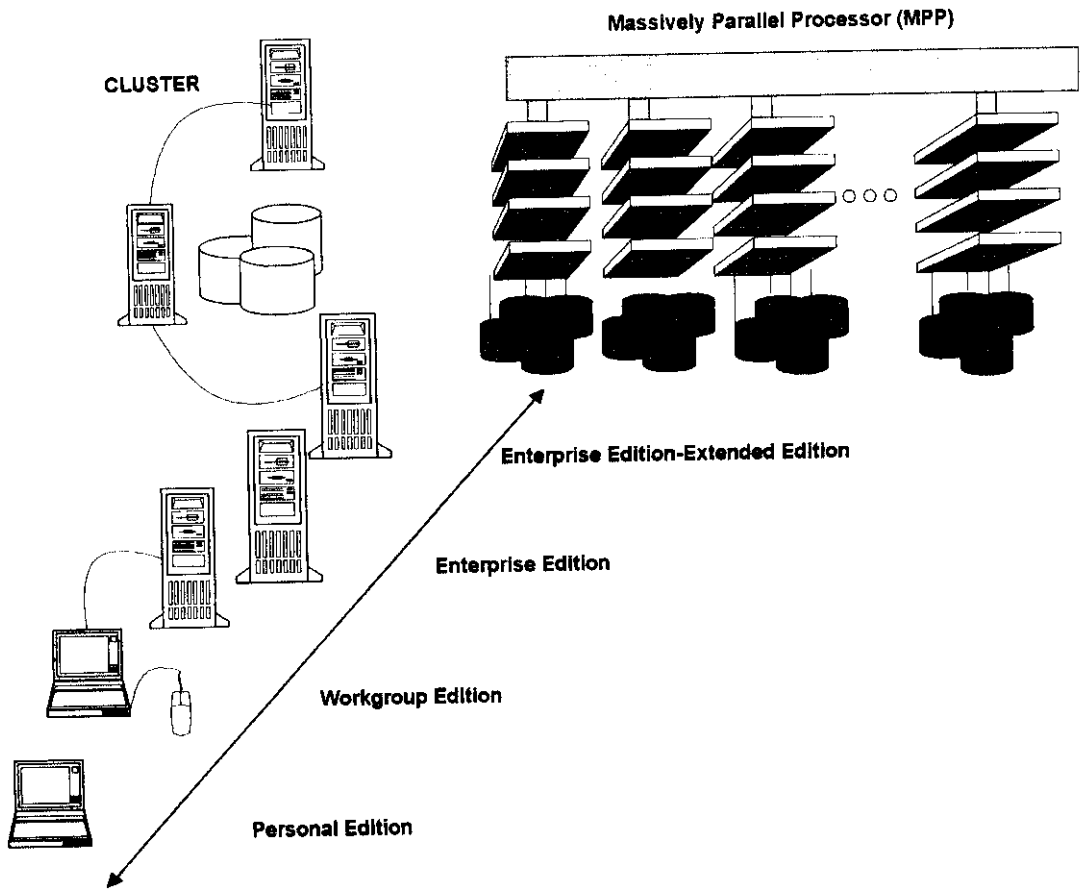


Fig. 1. DB2 Products

2.1.2 DB2 TOOLS

Las herramientas que están incluidas con el DB2 nos permiten ahorro de tiempo, un manejo más eficiente mediante un interfaz gráfico, amigable, así como también, la ejecución de comandos en un modo no interactivo.
Entre las principales tenemos:

- **Control Center:** es usado para administrar Servidores del DB2.
- **The Configuration Assistant:** es usado para configurar la comunicación entre Cliente y Servidor.

También existen herramientas básicas como son:

- **Wizards:** (Asistente) puede ser usado por usuarios principiantes y expertos, el cual nos ayuda a realizar una tarea específica paso a paso, como crear, respaldar y recuperar Base de Datos, crear tablas, etc.
- **Show SQL/Show Command:** Cuando una herramienta genera una sentencia SQL, esta opción nos muestra en una ventana la sentencia que se ha generado en la herramienta.

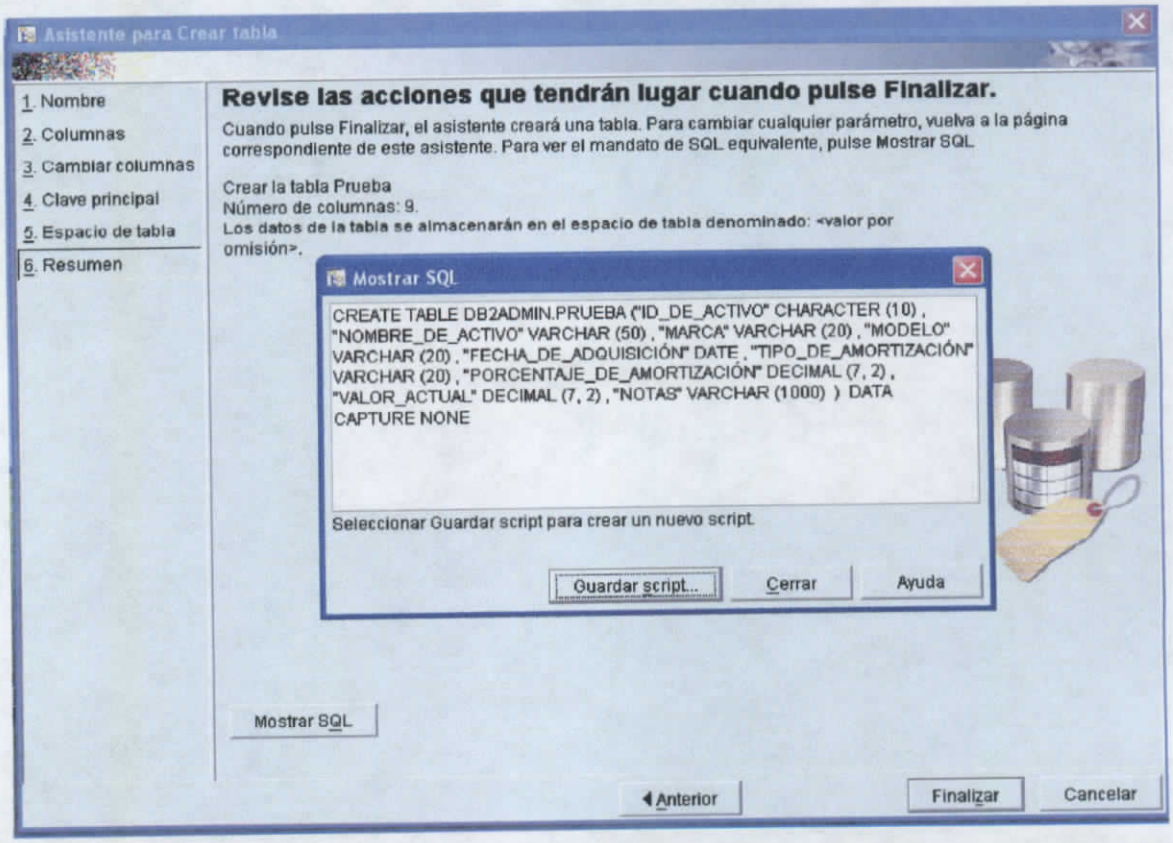


Fig. 2. DB2 Tools (Show SQL)

- **Show Related:** Esta herramienta muestra las relaciones entre tablas, Índices, Vistas, etc.

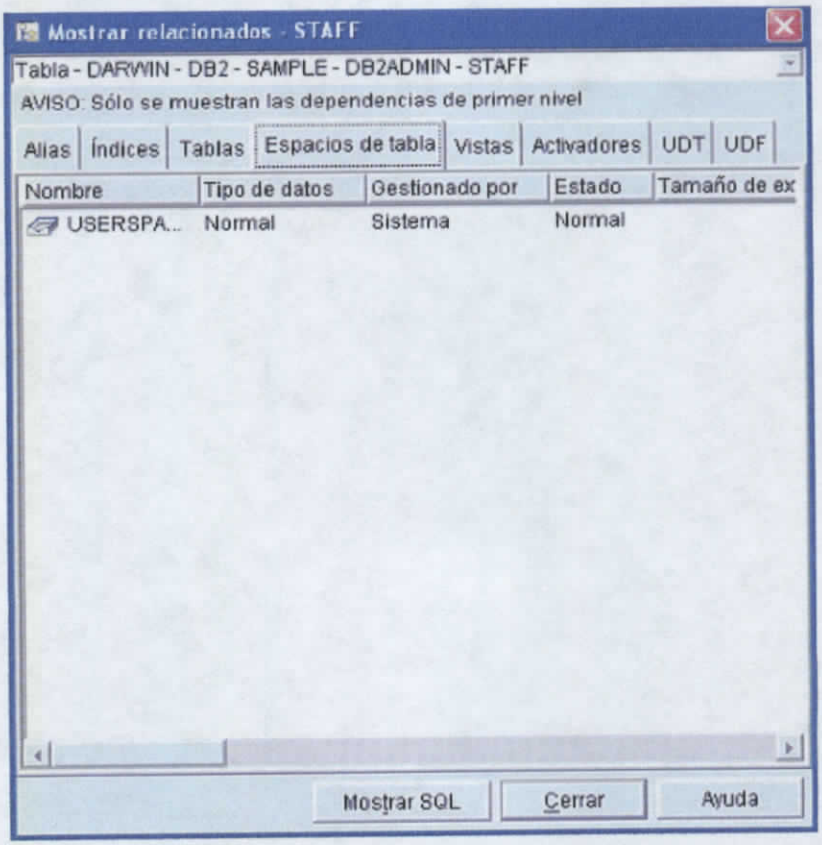


Fig. 3. DB2 Tools (Show Related)

2.1.3 CONTROL CENTER

El Centro de control proporciona una visión general y clara de todos los sistemas y objetos de la base de datos que se está gestionando. Puede también acceder a otras herramientas de administración desde el Centro de control; para ello seleccionamos un icono en la barra de herramientas del Centro de control.

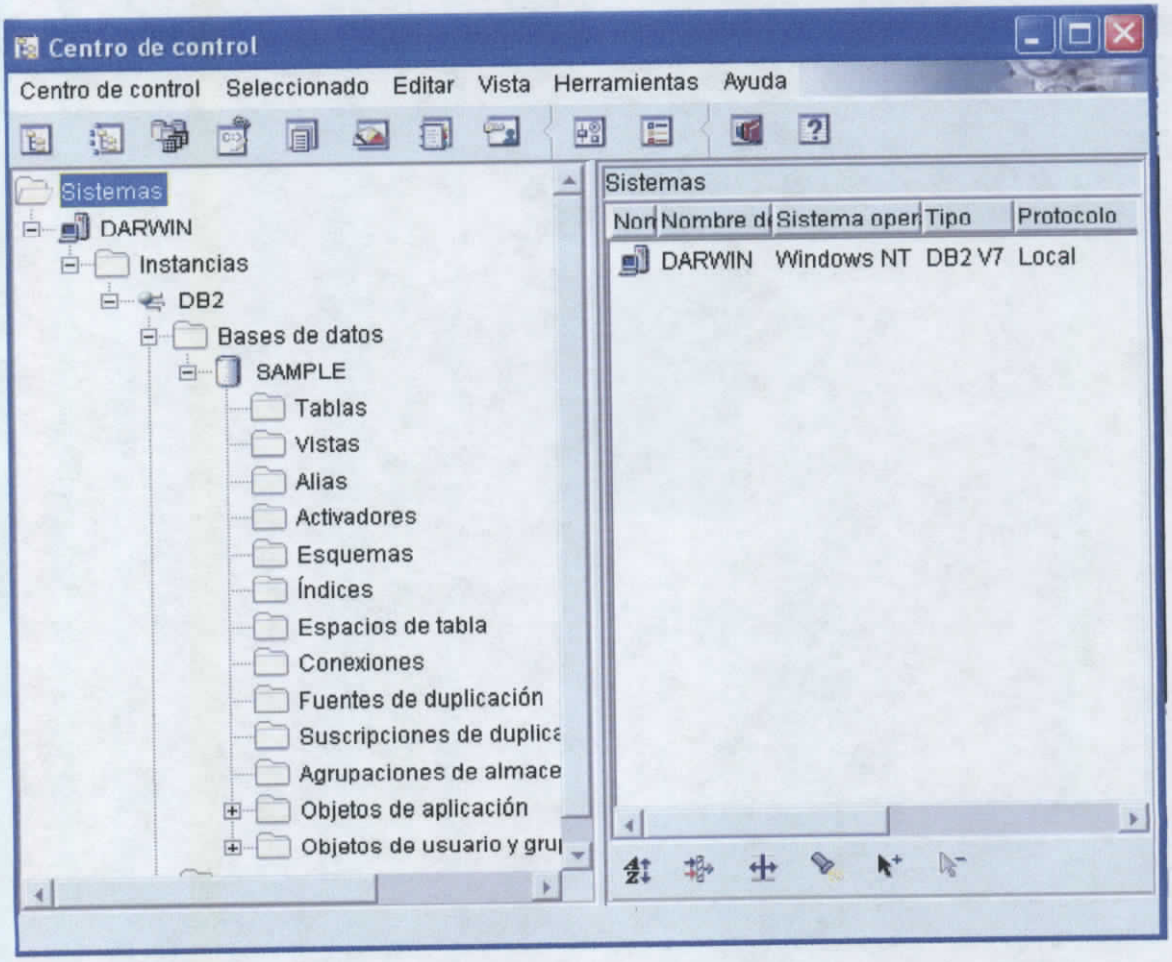


Fig. 4. Control Center

2.1.4 THE CONFIGURATION ASSISTANT

Esta herramienta nos permite mantener una lista de las bases de datos a las cuales nuestra aplicación puede conectarse, manejar y administrar.

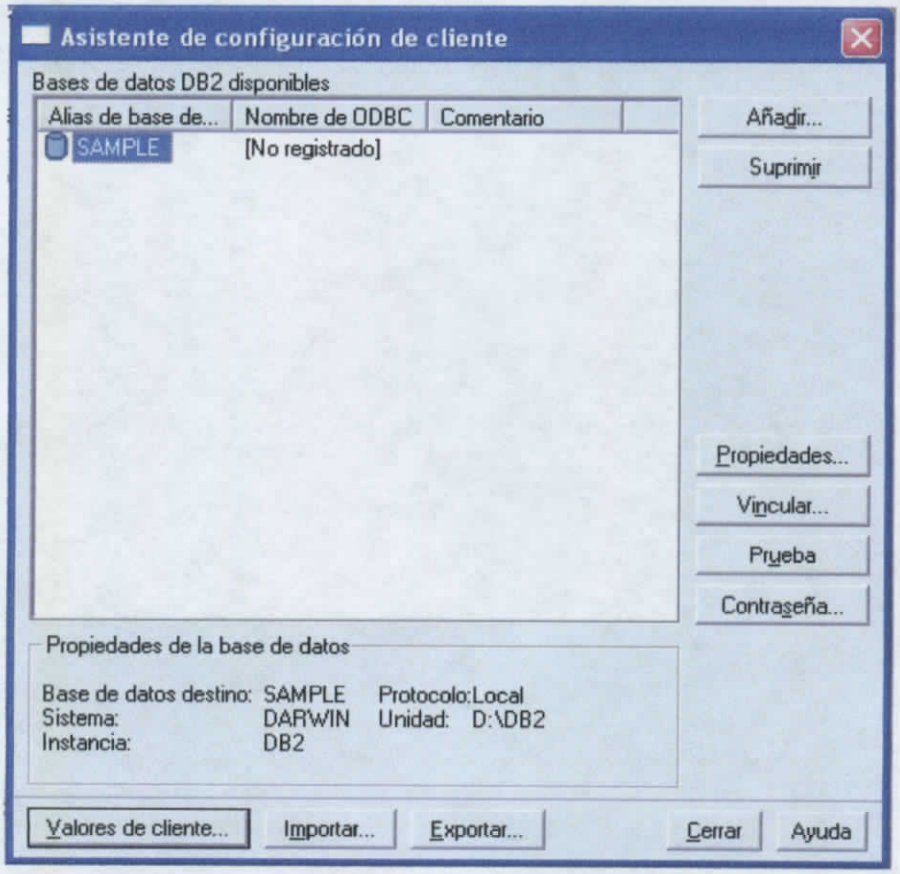


Fig. 5. The Configuration Assistant

2.1.5 OTHER DB2 TOOLS

Existen otras herramientas que vienen incluidas de manera gratuita en el DB2, estas herramientas son las siguientes:

- **Visual Explain**: le permite ver el plan de acceso para las sentencias de SQL explicadas como gráfico. Puede utilizar la información disponible del gráfico para ajustar las consultas de SQL para un mejor rendimiento.

Un gráfico de plan de acceso muestra detalles de:

- Tablas (y sus columnas asociadas) e índices
- Operadores (por ejemplo, uniones, clasificaciones y exploraciones de tabla)
- Funciones y espacios de tabla.

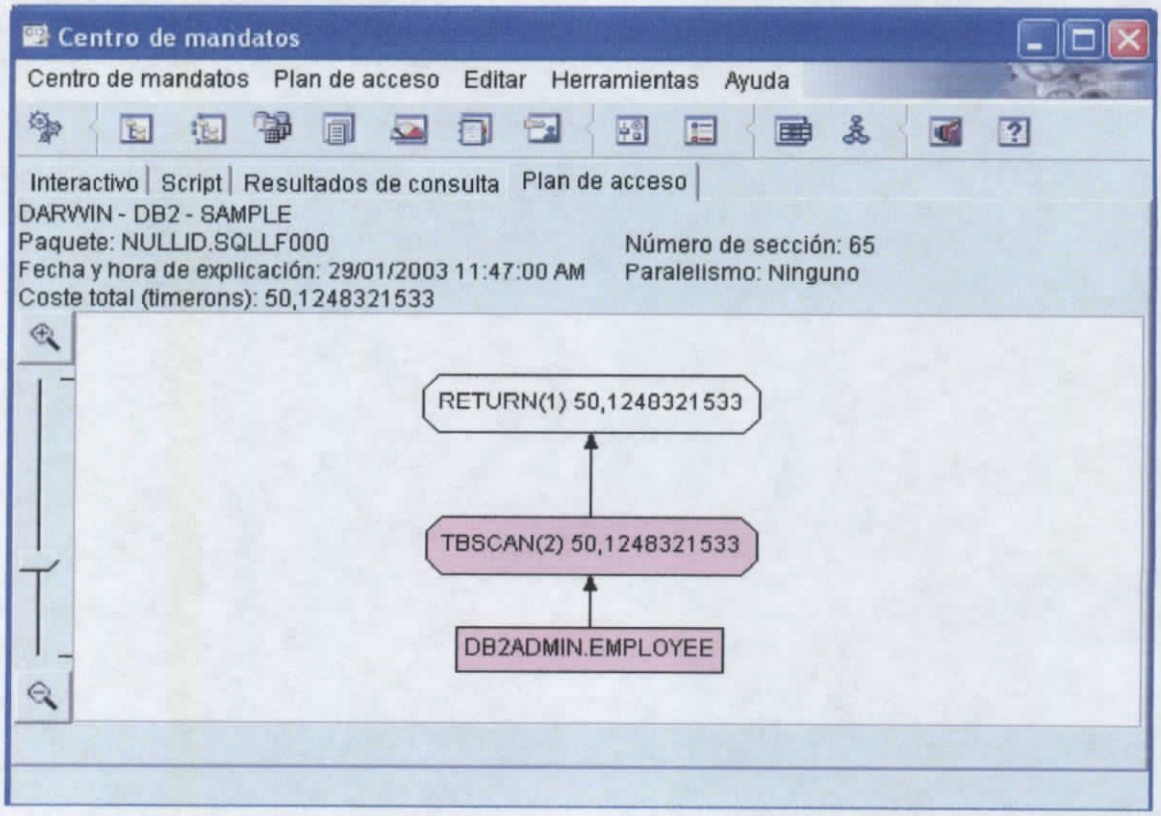


Fig. 6. Other DB2 Tools

- **Tool Setting:** Permite personalizar las herramientas gráficas y otras opciones del DB2 como: cambiar el tipo de letra para menú y texto.

2.2 DB2 SECURITY

La seguridad en las bases de datos es supremamente importante hoy en día. Las empresas podrían vender o adquirir productos a través del Internet, para lo cual el DB2 brinda la seguridad necesaria y confiable para las empresas.

La seguridad abarca los siguientes aspectos:

- Usuarios que están permitido acceder a las bases de datos.
- Dónde y cómo los passwords de los usuarios deben ser verificados.
- Los niveles de autoridad que un usuario posee.
- Los comandos que un usuario puede ejecutar.
- Los datos que un usuario esta permitido leer y/o alterar.
- Los objetos de las bases de datos que un usuario crea, altera, y/o elimina.

2.2.1 DB2 AUTHENTICATION

Permite un control de:

- Usuarios que pueden acceder a las bases de datos.
- Dónde y cómo los passwords de los usuarios deben ser verificados.

2.2.2 AUTHORITIES

Controla los siguientes aspectos en las seguridades de la base de datos:

- Los niveles de autoridad que un usuario posee.
- Los comandos que un usuario puede ejecutar.

- Los datos que un usuario esta permitido leer y/o alterar.
- Los objetos de las bases de datos que un usuario crea, altera, y/o elimina.

2.2.3 DB2 PRIVILEGES

Los privilegios pueden ser principalmente divididos en dos categorías:

- Privilegios a nivel de base de datos: son aquellos que el Administrador del Sistema (SYSADM) concede a un usuario para administrar la Base de Datos (DBA)
- Privilegios a nivel de objetos: son aquellos que concede el Administrador de la Base de Datos (DBA) a los usuarios para que puedan manipular los objetos existentes en la Base de Datos como son tablas, vistas, etc.

2.3 ACCESSING DB2 UDB DATA

DB2 nos brinda un acceso a la base de datos y a los objetos de una manera fácil a través de las herramientas que vienen incluidas en el programa con la cual el usuario puede hacer uso de acuerdo a su conveniencia.

2.3.1 WHAT MAKES UP A DB2 DATABASE

Las Bases de Datos son una colección de objetos con los cuales un usuario puede trabajar; tales como: tablas, vistas, índices, etc.

Una tabla es un conjunto de columnas y filas y todos los datos se encuentran almacenados en las tablas.

2.3.2 CREATING YOUR FIRST DATABASE

Una vez instalado el DB2, este nos permite instalar una base de datos de ejemplo, con la cual el usuario puede trabajar explotando así las características del DB2.

De igual manera el usuario puede crear su propia base de datos utilizando el asistente o a través de la ejecución de los comandos respectivos.

2.3.3 MANIPULATING DB2 OBJECTS WITH THE CONTROL CENTER

Esta herramienta nos permite desplegar todas las bases de datos y objetos que están creados en el DB2, de esta manera podemos trabajar de una manera interactiva con las tablas, vistas, índices, etc.

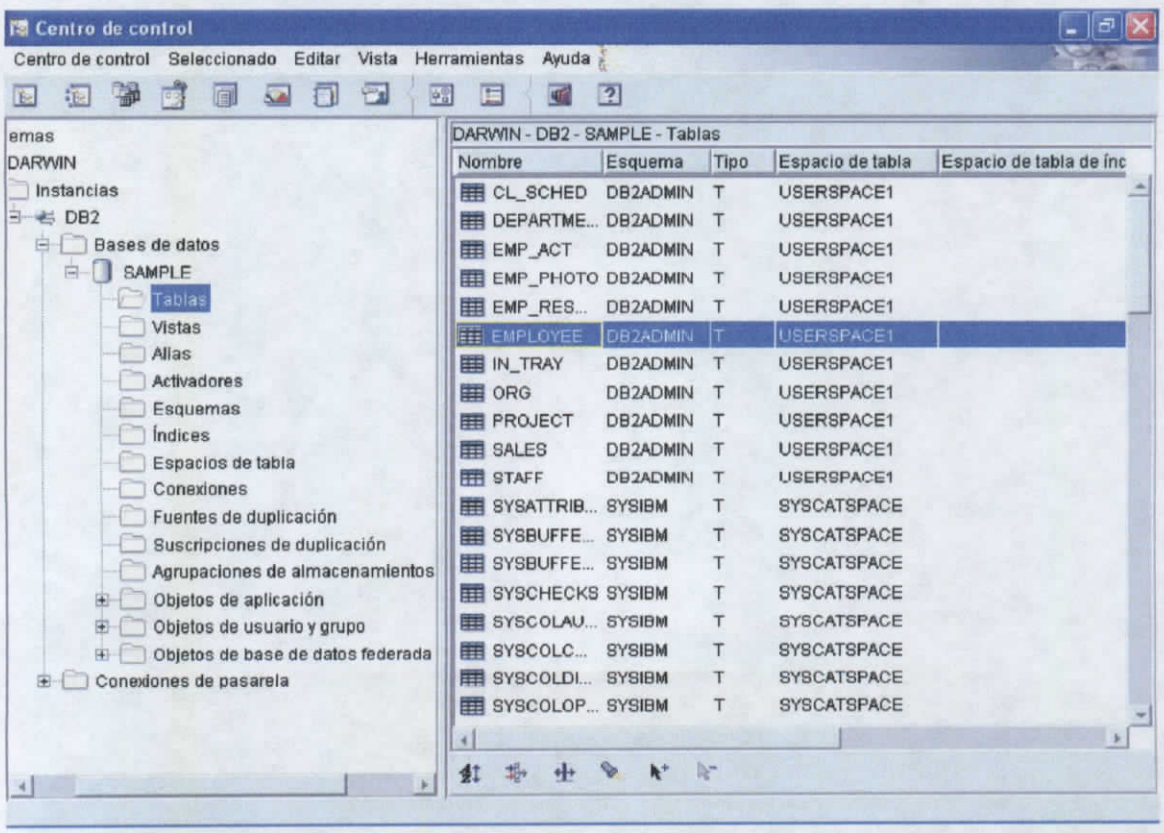


Fig. 7. Manipulating DB2 Objects with the Control Center

2.3.4 CREATING AND ACCESSING BASIC DB2 OBJECTS

Otras funciones que se pueden usar dentro del control center son: crear, modificar, borrar; la mayor parte de los objetos de la base de datos.

2.4 WORKING WITH DB2 UDB DATA

Este tema nos ayuda a dar un buen entendimiento de como el DB2 UDB usa el SQL para definir los objetos de las bases de datos como tablas, vistas e índices así como también manipular los datos.

2.4.1 STRUCTURED QUERY LANGUAGE (SQL)

SQL es un lenguaje usado para definir, manipular y controlar los objetos de la base de datos como: definir tablas, insertar datos dentro de las tablas, cambiar datos, sacar datos de las tablas.

2.4.2 DATA CONTROL LANGUAGE (DCL)

El DCL es usado para permitir el acceso y control de los objetos de la base de datos, a través del uso de las sentencias GRANT y REVOKE, el cual nos permite otorgar y revocar privilegios sobre los objetos de las bases de datos.

2.4.3 DATA DEFINITION LANGUAGE (DDL)

El DDL es usado para definir objetos dentro de la base de datos a través del uso de las sentencias CREATE, DROP, ALTER, el cual nos permite crear, eliminar o alterar.

2.4.4 DATA MANIPULATION LANGUAGE (DML)

El DML es usado para recuperar, insertar, actualizar o borrar datos almacenados dentro de las tablas o vistas en la base de datos a través del uso de las sentencias SELECT, INSERT, UPDATE, DELETE.

2.5 WORKING WITH DB2 UDB OBJECT

Este Tema habla acerca de los tipos de datos, definición de tablas, vistas e índices así como la manipulación de cada uno de ellos a través del uso del SQL, y como pueden ser usados en una aplicación.

2.5.1 DATA TYPE

DB2 nos proporciona una amplia variedad de tipos de datos dentro de los cuales se encuentran los básico como son: INTEGER, CHAR y DATE

Tipos de Datos Numéricos

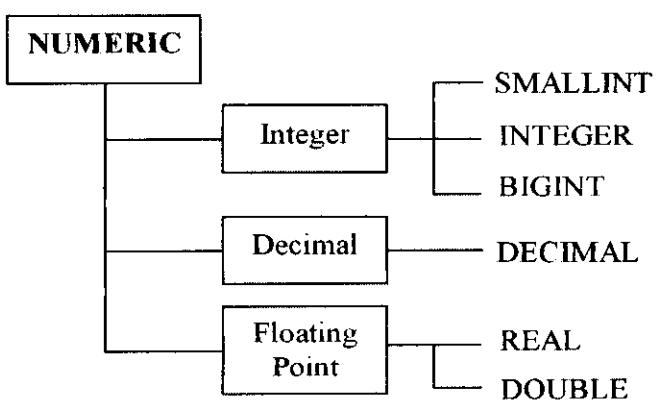


Fig. 8. Data Type (Numeric)

Tipos de datos Caracter

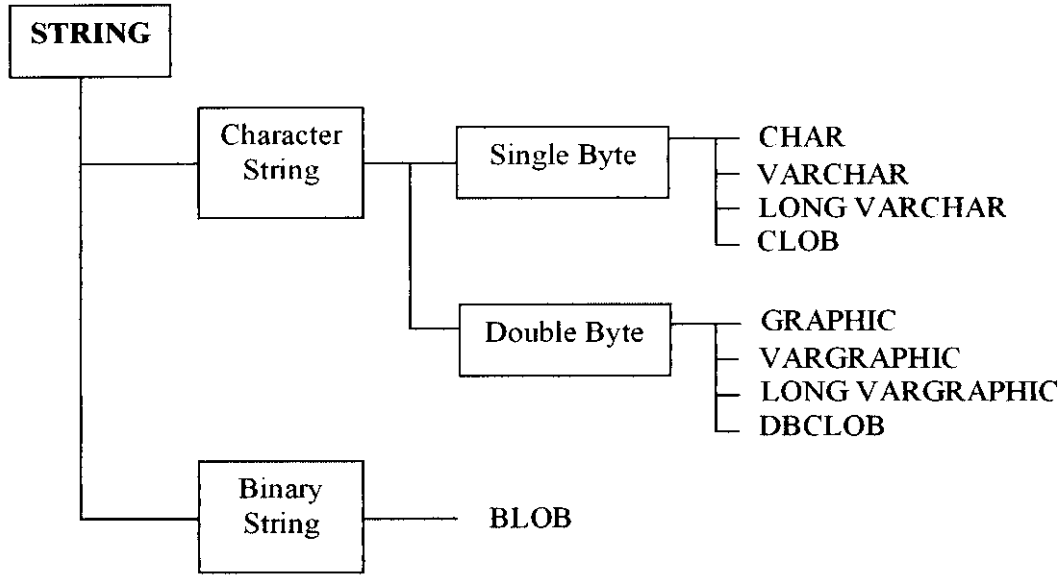


Fig. 9. Data Type (String)

Tipo de Datos Fecha

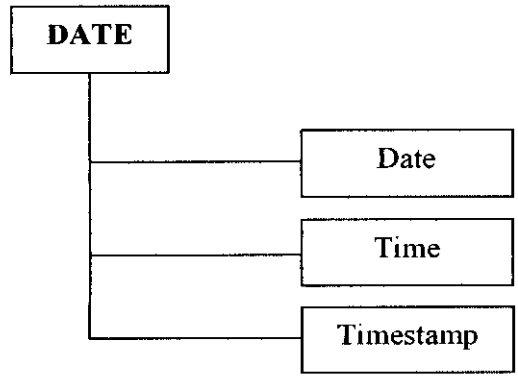


Fig. 10. Data Type (Date)

2.5.2 TABLES

Una tabla consiste en una o más columnas de varios tipos de datos. Los datos se encuentran almacenados en filas o records.

Nosotros podemos crear, alterar, eliminar tablas dentro de la base de datos mediante el uso de las sentencias CREATE, ALTER, DROP.

2.5.3 CONSTRAINTS

DB2 proporciona varias formas para controlar que dato puede ser almacenado en una columna. Estas características son llamadas constraints que ayuda al manejador de la base de datos a definir estas reglas sobre una columna o conjunto de columnas.

2.5.4 VIEWS

Las vistas permite a los usuarios o aplicaciones mirar los mismos datos en diferentes formas. Estas pueden ser usadas para restringir filas y columnas que pueden ser vistas o actualizadas por otros usuarios.

2.5.5 INDEXES

Un índice es una lista ordenada de valores de columna o columnas dentro de una tabla. Las cuales nos permite asegurar que los valores sean únicos en una columna o columnas, así como también mejora la búsqueda de datos.

2.6 CONCURRENCY

El estudio de la concurrencia nos ayuda a entender como varias aplicaciones acceden a ejecutar tareas de lectura y actualización de los datos en una tabla.

Es la habilidad del manejador de la base de datos para permitir a múltiples usuarios acceder a los datos de las tablas al mismo tiempo.

CAPITULO III

DESARROLLO DEL PROYECTO

PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

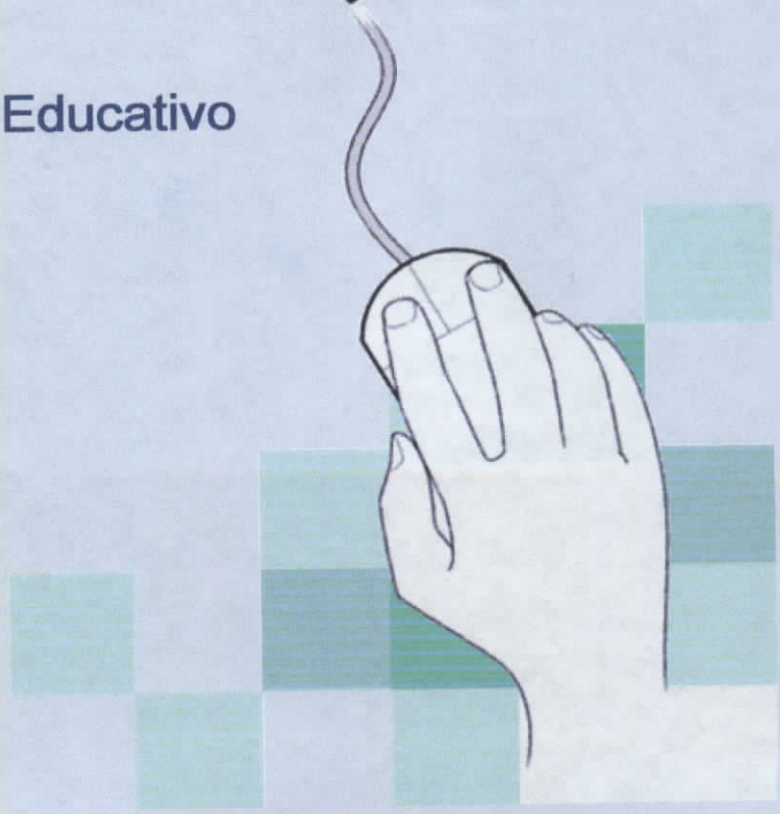
SEDE - AMBATO





IBM

Material Educativo





DB2 Family Fundamentals

Student Notebook

Material de Certificación

Contents

Units

Unit 1. Understanding A Table 1-1
Unit Objectives 1-2
1.1 Understanding A Table 1-3
The 'Ease' of working with a Relational Database 1-4
Find Phone Number of Larry Jones From Sales 1-5
The Problem With Hard Copy Data 1-6
Load Data in DBMS 1-7
Some of the Integrated Services DB2 UDB Provides 1-8
Concurrency – Down to the Row Level 1-9
Integrity 1-10
Security 1-11
DB2 UDB is a Relational Database Management Systems 1-12
The Language of Relational Database Management Systems 1-14
SQL Structure Query Language 1-15
Creating Tables in DB2 UDB – We Must Provide 1-16
Table Name Construction 1-17
Table and Column Name Conventions 1-18
Name Examples 1-19
Common Column Data Types 1-20
Sample Numeric Data Types 1-21
Sample Alphanumeric Data Types 1-22
Date Data Types 1-23
Time Data Types 1-24
Graphic String Format 1-25
Large Object – The Need 1-26
Create Table 1-27
DB2 Relational Extenders 1-28
NULL Characteristic 1-29
From an Editor that Talks to DB2 UDB 1-30
Unique Key, Primary Key 1-31
CREATE TABLE Statement 1-32
Foreign Key 1-33
Example CREATE TABLE Statement 1-34
Primary Keys, Foreign Key 1-35
A Look at Three Related Tables 1-36
The Rules DB2 Enforces to Maintain Referential Integrity 1-37
Adding Primary and Foreign Keys to an Existing table 1-38
Processing Rules Enforced to Maintain Referential Integrity 1-39
Primary Key Violation 1-40
Foreign Key Violation 1-41
Potential SYSADM Tasks 1-42

Potential DBADM Tasks 1-43
Potential Programmer Tasks 1-44
Potential End-User Tasks 1-45
Potential Operator Tasks 1-46
Unit Summary 1-47
Checkpoint 1-48

Unit 2. How Does An End-User Use DB2 UDB? 2-1
Unit Objectives 2-2
2.1 Ways to Run SQL 2-3
How Do I 'Talk' to DB2 UDB 2-4
Command Line Processor – Interactive 2-5
Command Center 2-6
2.2 Writing SQL 2-7
Who Is an End-User? 2-8
Running Programs that 'Talk' to DB2 UDB 2-9
And If I'm Writing My Own SQL? 2-10
Four SELECT Statement Clauses 2-11
To Retrieve All Columns for All Rows 2-12
To SELECT Specified Rows 2-13
Base Table 2-14
Conceptual Intermediate Table 2-15
Numeric Comparisons 2-16
SQL Comparisons Alternatives 2-17
Function – Two Types 2-18
Column Functions 2-19
Scalar Function 2-20
UPPER 2-21
Example SELECT Statement 2-22
Conceptual Execution of a SELECT 2-23
Conceptual Execution of a SELECT (Cont) 2-24
SELECT Clause – Column Naming 2-25
Inserting Rows into the Table 2-26
Update Columns 2-27
Delete Rows 2-28
Delete Rules 2-29
Delete Rules – ON DELETE CASCADE 2-30
Delete Rules – ON DELETE SET NULL 2-31
Delete Rules – ON DELETE RESTRICT 2-32
Unit Summary 2-33
Checkpoint 2-34

Unit 3. What does an Administrator do in DB2 UDB? 3-1
Unit Objectives 3-2
3.1 What Does A System Administrator Do in DB2 UDB? 3-3
What does a System Administrator Do in DB2 UDB? 3-4
IBM DB2 UDB Family 3-5

DB2 UDB Runs in a Massively Parallel Processor environment..... 3-7
Why massively Parallel processors for VLDBs 3-8
How can we fill the pool faster? 3-9
What happens when we have to clean the pool? 3-10
Possible option – how about 209 pools? 3-11
Installation – physica install 3-12
Create system level object 3-13
Using the DB2 UDB control center 3-14
Grant privileges 3-15
3.2 What does a database administrator do in DB" UDB? 3-16
What does a database administrator do in DB2 UDB?..... 3-17
Creating tables using the table wizard 3-18
Name the table 3-19
Choose columns 3-20
Defining a primary key 3-21
Defining a foreign key 3-22
Review SQL – Finish 3-23
Alter table 3-24
Add business rules, foreing key, and so on 3-25
3.3 Indexes 3-26
Table without indexes – from the table company. directory 3-27
An Index is 3-28
Creating Index 3-29
Creating an Index 3-30
3.4 Views..... 3-31
Views 3-32
Creating a view 3-33
Using a view 3-34
Updating a through a view 3-35
3.5 Drop..... 3-36
The DBA cleans up 3-37
Clean up using the GUI 3-38
DROP dependencies 3-39
3.6 Locking 3-40
Concurrency – table level (1 of 6) 3-41
Concurrency – table level (2 of 6) 3-42
Concurrency – table level (3 of 6) 3-43
Concurrency – table level (4 of 6) 3-44
Concurrency – table level (5 of 6) 3-45
Concurrency – table level (6 of 6)..... 3-46
Concurrency – Row level 3-47
3.7 Security 3-48
To get to DB2 UDB 3-49
DB2 security 3-50
Privileges 3-51
Owner's (implicit) table privileges 3-52
Explicit table privileges 3-53
Indirect privileges – group 3-54
Administrative authorities 3-55

Views and security 3-56
Revoke 3-57
Unit Summary 3-58
Checkpoint 3-59

Unit 4. Simple SQL Queries 4-1
Unit Objectives 4-2
4.1 Simple SQL Queries 4-3
EMPLOYEE Table (Part 1) 4-4
EMPLOYEE Table (Part 1 Cont'd) 4-5
EMPLOYEE Table (Part 2) 4-6
EMPLOYEE Table (Part 2 Cont'd) 4-7
DEPARTMENT Table 4-8
PROJECT Table 4-9
Structure of an SQL Query 4-10
Selecting all Columns 4-11
Selecting Specific Columns 4-12
Select with Ordered Output 4-13
Select with Ordered Output (Cont'd) 4-14
Alternate Specifications 4-15
Suppressing Duplicate Output Rows 4-16
Suppressing Duplicate Output Rows (Cont'd) 4-17
Selecting Rows by Character Data 4-18
Selecting Rows by Numerical Data 4-19
Comparison Operators 4-20
AND and OR - Principle 4-21
Multiple Conditions - AND 4-22
Multiple Conditions - OR 4-23
Multiple Conditions - AND/OR 4-24
Multiple Conditions - Parentheses 4-25
SELECT with IN 4-26
SELECT with BETWEEN 4-27
Selecting NULL Values 4-29
Partial Search - LIKE 4-30
Partial Search - Examples with '%' 4-31
Partial Search - Example with '_' 4-32
Negation 4-33
Unit Summary 4-34
Checkpoint 4-35

Unit 5. Simple Retrieving Data from Multiple Tables 5-1
Unit Objectives 5-2
4.1 Accessing Data Stored in Multiple Tables 5-3
Traditional Filing 5-4
Retrieving Data from Multiple Tables (Principle) 5-5
Retrieving Data from Multiple Tables (JOIN) 5-6
JOIN Syntax 1 5-7
JOIN Syntax 2 (JOIN Keyword) 5-8
JOIN with Three Tables 5-9

JOIN with Three Tables (Cont'd) 5-10
Correlation Name 5-11
Joining a Table with Itself 5-12
Joining a Table with Itself (Cont'd) 5-13
Unit Summary 5-14
Checkpoint 5-15

Unit 6. Scalar Functions and Arithmetic 6-1
Unit Objectives 6-2
6.1 Scalar Functions and Arithmetic 6-3
Select Calculated Values 6-4
Naming Result Columns 6-5
Substitution of NULL Values 6-6
Arithmetic with NULL Values 6-7
Calculated Values 6-8
Decimal Representation of a Value 6-9
Decimal Values - Truncation and Rounding 6-10
Condition on Calculated Values 6-11
Date and Time 6-12
Comparison with Dates 6-13
DATE/TIME Arithmetic 6-14
Subtraction of Dates 6-15
Date/Time Scalar Functions 6-16
Date Scalar Functions 6-17
DATE Arithmetic 6-18
DATE Arithmetic (Cont'd) 6-19
Substring of Strings 6-20
Substring of Strings (Cont'd) 6-21
Concatenation of Values 6-22
Case Expression 6-23
Case Expression (Cont'd) 6-24
Unit Summary 6-25
Checkpoint 6-26

Unit 7. Column Functions and Grouping 7-1
Unit Objectives 7-2
7.1 Column Functions and Grouping 7-3
Column Functions 7-4
Sample Column Functions 7-5
Column Functions Based on Subset 7-6
GROUP BY 7-7
GROUP BY – HAVING 7-8
GROUP BY - HAVING (Cont'd) 7-9
Examples with HAVING 7-10
Examples with HAVING (Cont'd) 7-11
Examples with HAVING (Cont'd) 7-12
Unit Summary 7-13
Checkpoint 7-14

Unit 8. UNION 8-1
Unit Objectives 8-2
8.1 UNION 8-3
UNION Principle 8-4
Combination of Multiple Result Tables 8-5
UNION Details 8-6
UNION Example 8-7
Generation of Fitting Result Rows 8-8
UNION ALL-Principle 8-9
UNION ALL Example 8-10
Unit Summary 8-11
Checkpoint 8-12

Unit 9. Using Subqueries 9-1
Unit Objectives 9-2
9.1 Using Subqueries 9-3
Result Using Separate Selects 9-4
Same Result Using a Subquery 9-5
Subquery with Basic Predicates 9-6
Subquery - Set of Values 9-7
Subquery Using IN 9-8
Subquery Using ALL 9-10
Subqueries Using ANY or SOME 9-11
Subquery within Having Clause 9-12
Unit Summary 9-13
Checkpoint 9-14

Unit 10. Maintaining Data 10-1
Unit Objectives 10-2
10.1 Maintaining Data 10-3
TESTEMP Table 10-4
Create Table TESTEMP 10-5
Create View PERSINFO 10-6
Insert a Single Row 10-7
Insert Multiple Rows 10-8
Update Columns 10-9
Update Columns (Cont'd) 10-10
Delete Rows 10-11
Drop Table TESTEMP 10-12
Unit Summary 10-13
Checkpoint 10-14

Appendix A. Checkpoint Solutions A-1

Unit 1 Understanding A Table

What This Unit Is About

This unit introduces the DB2 UDB Database Manager and the basic concepts behind a table.

What You Should Be Able to Do

After completing this unit, you should be able to:

- List three services provided by a database manager
- Define a relation
- Name the language used to "talk" to a relational database manager (RDBM)
- List three characteristics assigned to each column
- List several data types that can be assigned to a column
- List three forms of key used in a DB2 UDB table.

How You Will Check Your Progress

Accountability:

- Checkpoint questions

Unit objectives

After completing this unit, you should be able to:

- List three services provided by a database manager
- Define a RELATION
- Name the language used to "Talk" to a relational database manager (RBDM)
- List three characteristics assigned to each column
- List several data types that can be assigned to a column
- List three forms of key used in DB2 UDB tables

Figure 1-1 . Unit Objectives

Notes:

Database managers are applications that maintain our data. Each database manager provides a set of automated services. We will discuss three of the more common services.

The word relation is simply a synonym for the word table.

Relational database manager understand the SQL language

A column in a table can have many characteristic assigned to it. We will discuss three of the characteristics that are assigned to all columns.

We will discuss the most common DB2 UDB data types.

The three keys are: unique key, primary key, and foreign key.

The 'Ease' of working With a Relational Database



Figure 1-2 The 'Ease' of working with a Relational Database

Notes:

We work with tables of data everyday. A phone book is a table of data. Each page has the same columns (name, address and phone number) just different rows. Unlike tables of data in a relational database manager, phone books are in alphabetical sequence.

Find Phone number of Larry Jones from sales



LAST NAME	FIRST NAME	MI	DEPT	PHONE
JONES	ANDY	B	SALES	6-3357
JONES	ANN	S	ACC	6-9271
JONES	BRENDA	L	OPER	6-8956
JONES	FRANK	D	ENG	6-7224
JONES	LARRY	R	SALES	6-4367
JONES	LARRY	R	SALES	6-4588
JONES	SANDRA	D	OPER	6-8499
JONES	SANDY		REPRO	6-2875

Figure 1-3 Find Phone number of Larry Jones from sales

Notes:

If there are multiple rows (more than one Larry Jones from sales) that meet our search criteria we must return multiple phone numbers.

The Problem With Hard Copy Data

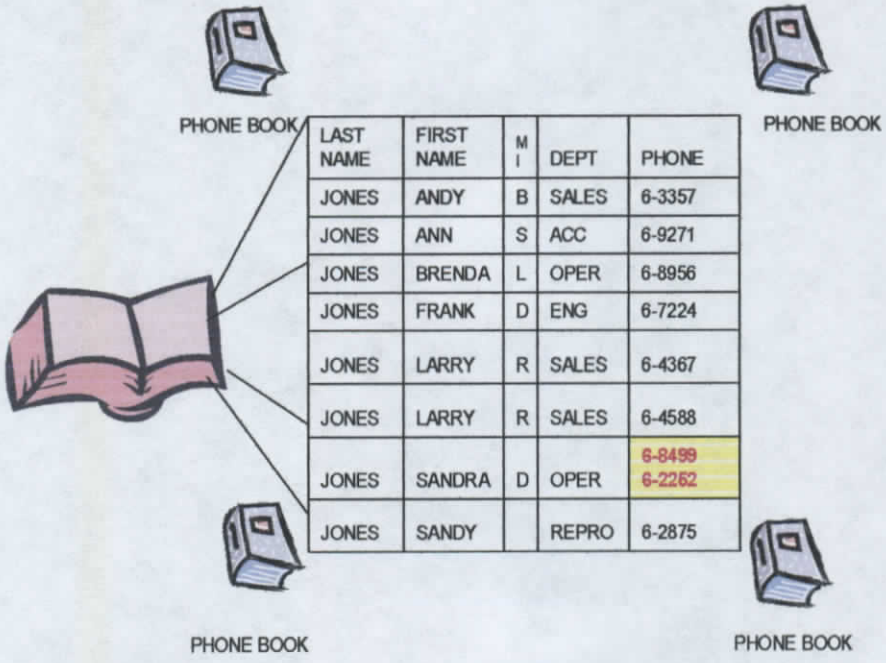


Figure 1-4 The Problem With Hard Copy Data

Notes:

Phone books are updated and issued periodically
Corrections made to one copy of the book are not distributed to the other copies around the company.

New listings and changes must be 'discovered' by everyone.

Load data in DBMS

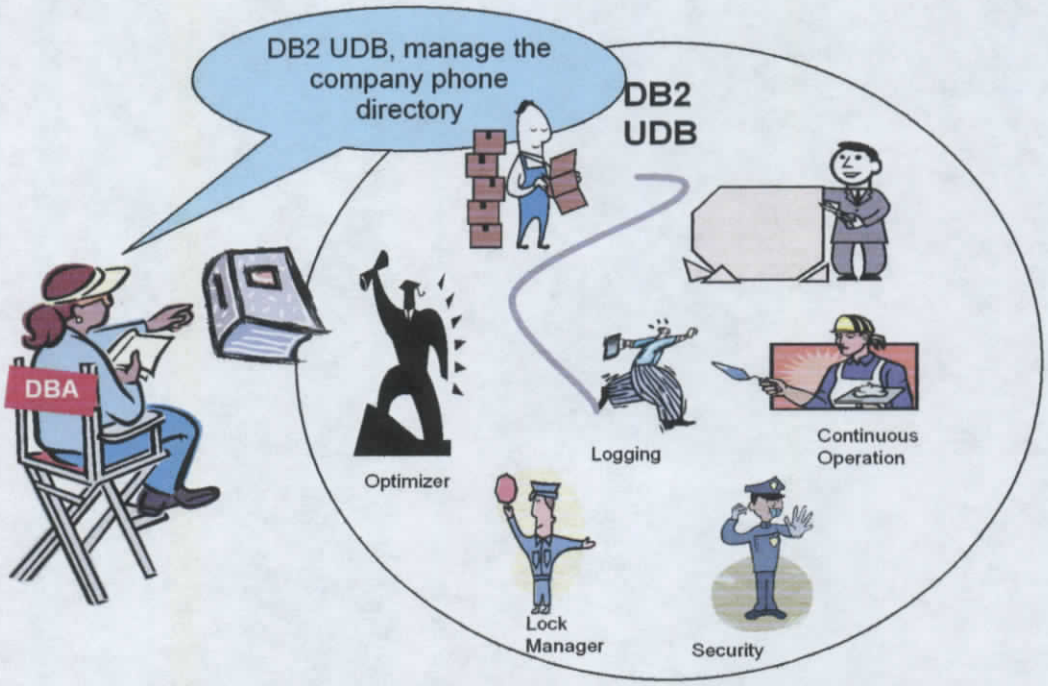


Figure 1-5. Load Data in DBMS

Notes:

Database managers provide many services. These are but a few.

- Logging
- Security
- Optimization
- Locking
- Recovery
- Data Integrity

Some of the Integrated Services DB2 UDB Provides

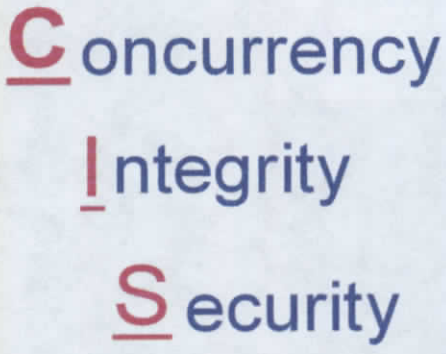
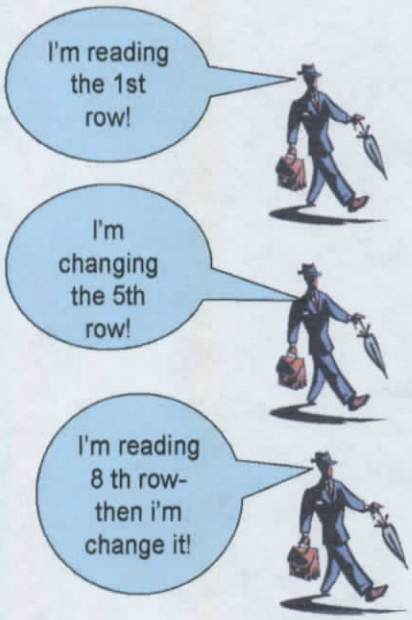


Figure 1-6 Some of the Integrated Services DB2 UDB Provides

Notes:

The first 'C' in the word concurrency is pronounced with a k sound. You can use the acronym CIS (pronounced kiss) to help you remember these services (the kissing services).

Concurrency – Down to the Row Level



LAST NAME	FIRST NAME	MI	DEPT	PHONE
JONES	ANDY	B	SALES	6-3357
JONES	ANN	S	ACC	6-9271
JONES	BRENDA	L	OPER	6-8956
JONES	FRANK	D	ENG	6-7224
JONES	LARRY	R	SALES	6-4367
JONES	LARRY	R	SALES	6-4588
JONES	SANDRA	D	OPER	6-8499
JONES	SANDY		REPRO	6-2875

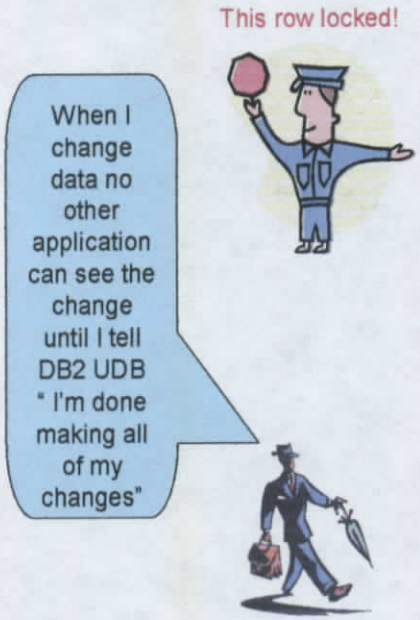
A page of data

Figure 1-7 Concurrency – Down to the Row Level

Notes:

Concurrency is the database manager's ability to allow multiple users access to a table's data, at the same time.

Integrity



LAST NAME	FIRST NAME	MI	DEPT	PHONE
JONES	ANDY	B	SALES	6-3357
JONES	ANN	S	ACC	6-9271
JONES	BRENDA	L	OPER	6-8956
JONES	FRANK	D	ENG	6-7224
JONES	LARRY	R	SALES	6-4367
JONES	LARRY	R	SALES	6-4588
JONES	SANDRA	D	OPER	6-8499
JONES	SANDY		REPRO	6-2875

Figure 1-8 Integrity

Notes:

A database manager ensures that while a row is in the process of being changed, no other application can access the row until the change process is completed.

The database manager also ensures that if an application is changing a set of rows, all rows in the set will be changed, or none of the rows will be changed.

Security

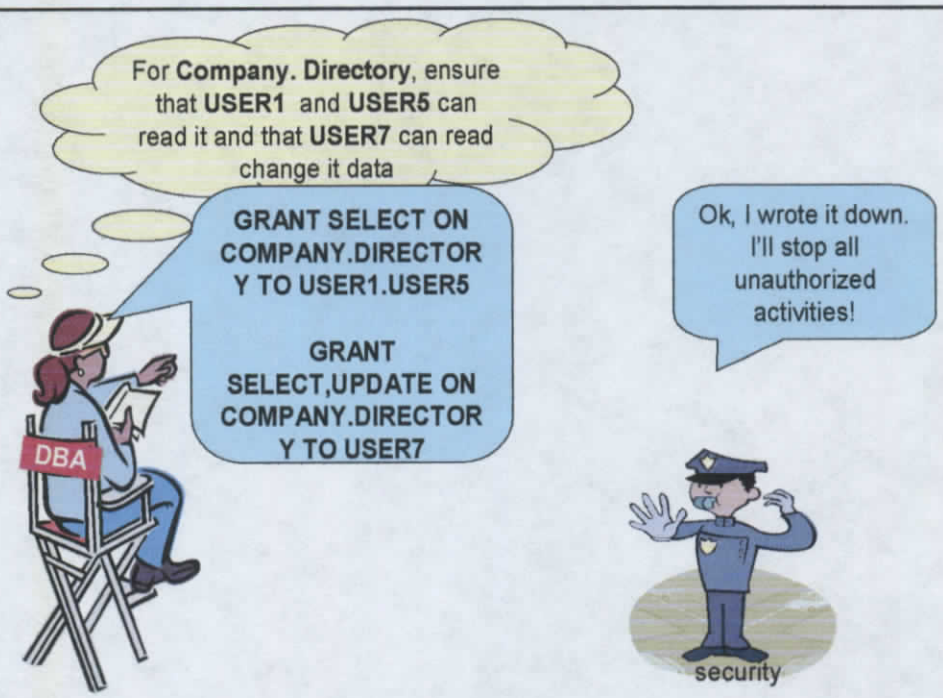


Figure 1-9 Security

Notes:

Authorizations are given with GRANT SQL statements.

The DBM catalogs the authorizations in catalog tables.

Each time an SQL statement is executed the database manages (DBM) will check the authorization catalog tables to ensure that the ID executing the statement is authorized to do so.

DB2 UDB Is a Relational Database Manager System

" RELATION " = TABLE

(Data stored in columns and rows)

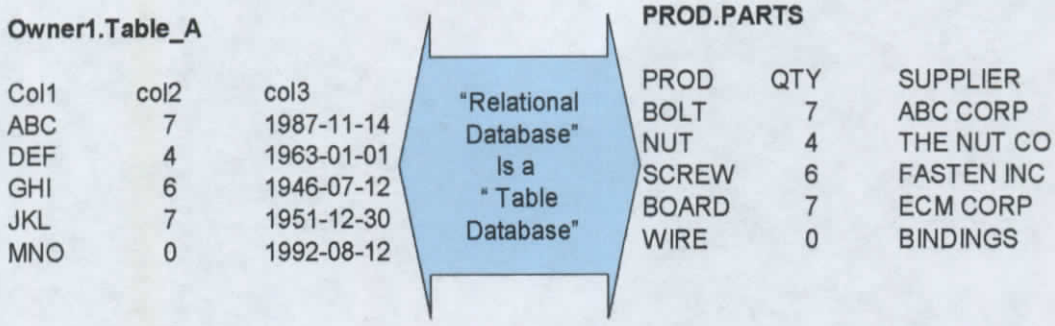


Figure 1-10 DB2 UDB Is a Relational Database Manager System

Notes:

The database management system we are working with in class is DB2 UDB.

DB2 UDB is a relational database management system (RDBMS). The word relation is a synonym for the word table. A relational database management system is a database manager that manages data perceived to be stored in a table format, that is, columns and rows.

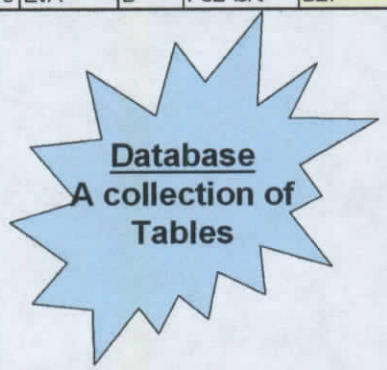
Relational Database Management System

EMPLOYEE

EMPNO	FIRSTNME	MIDINT	LASTANAME	WORKDEP	...
000010	CHRISTINE	I	HAAS	A00	
000020	MICHAEL	L	THOMPSON	B01	
000030	SALLY	A	KWAN	C01	
000050	JOHN	B	GEYER	E01	
000060	IRVING	F	STERN	D11	
000070	EVA	D	PULASKI	D21	

DEPARTMENT

DEPTNO	DEPTNAME	...
A00	SPIFFY COMPUTER SERVICE DIV.	
B01	PLANNING	
C01	INFORMATION CENTER	
D01	DEVELOPMENT CENTER	
D11	MANUFACTURING SYSTEMS	
D21	ADMINISTRATION SYSTEM	



PROJECT

PROJNO	PROJNAME	DEPTNO	...
AD3100	ADMIN SERVICE	D01	
AD3110	GENERAL ADMIN SYSTEM	D21	
AD3111	PAYROLL PROGRAMMING	D21	
AD3112	PERSONNEL PROGRAMMING	D21	
AD3113	ACCOUNT PROGRAMMING	D21	
IF1000	QUERY SERVICES	C01	

Figure 1-11 Relational Database Management System

Notes:

A relational database is a collection of tables.

A table is a set of columns and rows. All data is stored in tables.

Every table has a name.

In a well-designed database, each table contains data about a specific subject (entity); for instance, one table contains employee data, another table department data, and a third one data about projects.

The data base management system manages the data.

If someone requires information about an employee, he/she must talk to the data management system in a language which the system understands: SQL – Structured Query Language.

The Language of Relational Database Management System

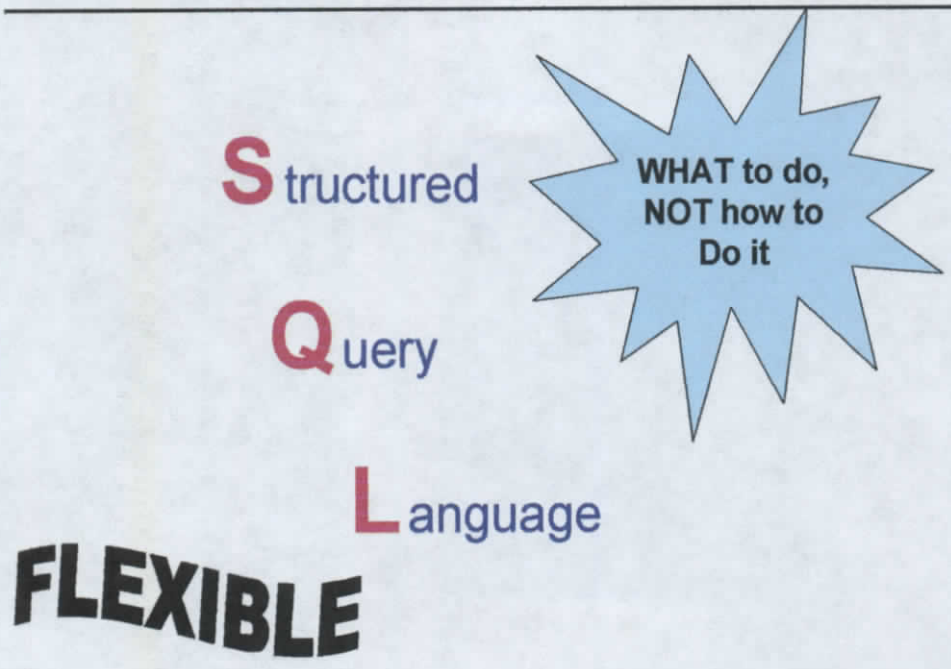


Figure 1-12 The language of Relational Database Management System

Notes:

SQL is a non-navigational language. With SQL we tell DB2 UDB what to do, not how to do it.

SQL Structure Query Language

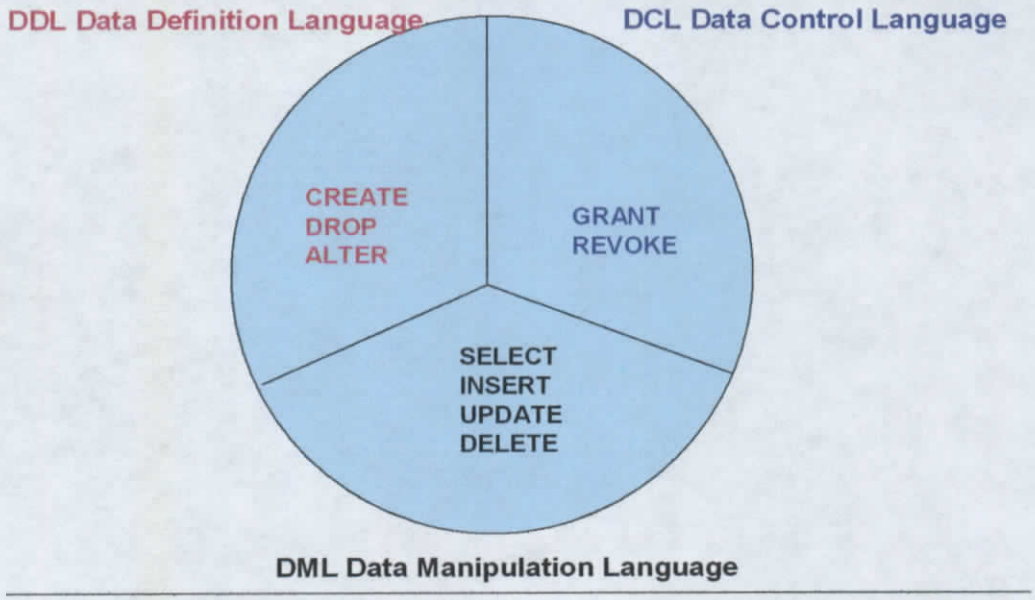


Figure 1-13 SQL Structure Query Language

Notes:

SQL has similarities with the human language. By means of SQL, we communicate with the RDBMS. We tell DB2 UDB what we want, not how DB2 UDB should access the data.

SQL can be used either interactively or embedded in an application program, for example, COBOL, PL/1, or C.

The object of a DML query is a set of table. The result of the DML query is a set of qualifying rows from these tables.

- DDL is mostly used by data base administrators to define DB2 UDB objects such as tables, views and indices
- DML is used by all data base users to retrieve information or to maintain data
- DCL is also used by data base administrators to allow or disallow activities in DB2 UDB by grating or revoking privileges

Most Relational Database Managers understand SQL.

Creating Tables in DB2 UDB – We Must Provide...

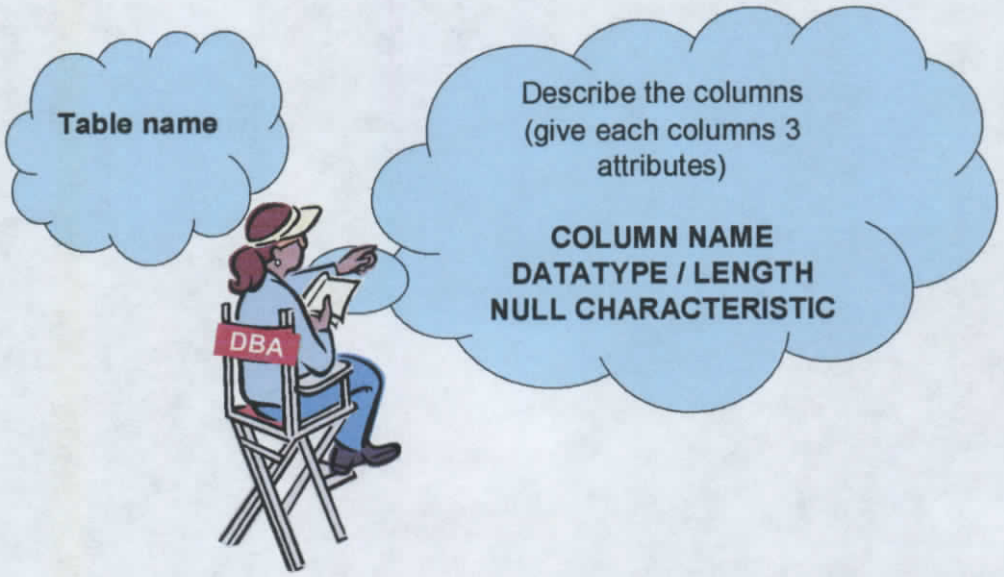


Figure 1-14 Creating Tables in DB2 UDB – We Must Provide...

Notes:

Every table must be given a name. Each of the table's columns must be assigned three attributes:

- Name
- Data type / length
- Null Characteristic

Table Name Construction

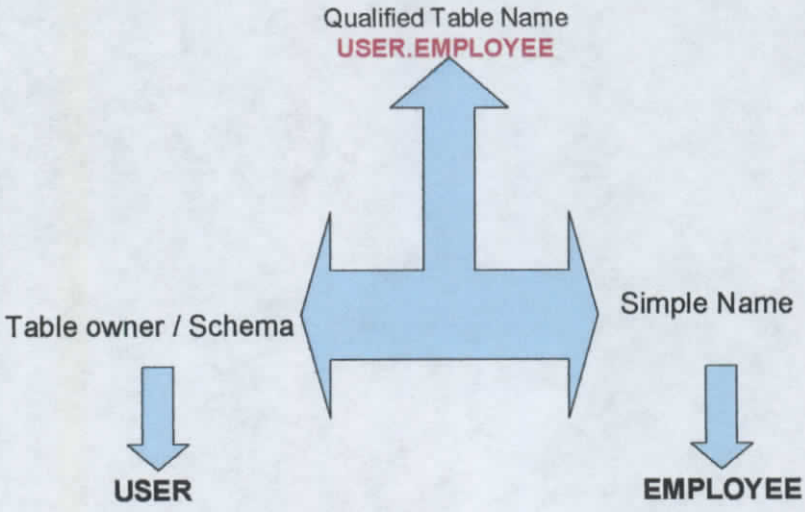


Figure 1-15 Table Name Construction

Notes:

- Every table must have a unique name.
- Every table name consists of at least two parts referred to as simple table name and its qualifier (tables used in a distributed processing environment may have three part names).
- The qualifier precedes the simple table name and is separated from it by a period

The simple table name must be unique within a qualifier.

Table and Column Naming Conventions

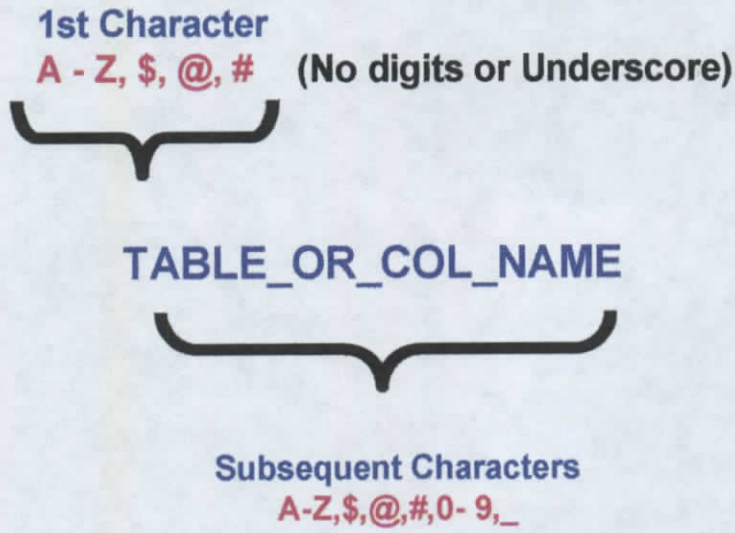


Figure 1-16 Table and Column Naming Conventions

Notes:

Table and column names must begin with an alpha character. An alpha character is in the set (A-Z, $, @, #). The remaining characters can be alpha, numeric (digits 0-9). and the underscore (_). Table and column names can be from 1 to at least 18 characters long. Newer releases of DB2 UDB support longer table and column names.

Name Examples

TABLE_3	SQL DIVIDER CHAR
SOC_SEC_#	ALPHA CHARACTER
"SAL+COM"	SPECIAL CHAR, NAME DELIMITER
"UPDATE"	RESERVED WORD NAME DELIMITER

Figure 1-17 Name Examples

Notes:

When non – standard characters are used in table or column names, within our queries the names must be enclosed in double quotes.

Common Column Data Type

Character Data	**CHAR(X)** **VARCHAR(X)** **LONGVARCHAR**
Numeric Data	**SMALLINT** **INTEGER** **DEC(X,Y)**
Date/Time	**DATE** **TIME** **TIMESTAMP**

Figure 1-18 Common Column Data Type

Notes:

Date, times and timestamps are stored internally as numeric values but must be treated as character string in an SQL statement.

Sample Numeric Data Types

Data Type	Comment
SMALLINT	Whole number from -32768 to +32767 (16 Bits)
INTEGER	Whole number from -2,147,483,648 to + 2,147,483,647 (32 Bits)
DECIMAL (m, n)	Decimal numbers with max. 31 digits m=total number of digits n= total of decimal digits

Figure 1-19 Sample Numeric Data Types

Notes:

The visual shows a few of the DB2 UDB supported numeric data types. For complete list of supported data types in your environment.

Sample Alphanumeric Data Type

Data Type	Comment
CHAR (n) or CHARACTER (n)	String (Fixed length) (1-254) bytes
VARCHAR (n)	STRING (Variable length) (1-32,672) bytes

Figure 1-20 Sample Alphanumeric Data Type

Notes:

The visual shows a few of the alphanumeric data types available for the various platforms. For completeness, see the reference manual for your environment.

Alphanumeric data may contain any character or printable special character.

In queries, alphanumeric values must be enclosed in single quotes.

Date Data Type

Data Type	Comment	
DATE	Display / Input Format	
YYYYMMDD	USA	mm/dd/yyyy
	EUR	dd.mm.yyyy
	ISO	yyyy-mm-dd
	JIS	yyyy-mm-dd

Figure 1-21 Date Data Type

Notes:

USA = US Standard
EUR = European Standard
ISO = International Standards Organization (Default)
JIS = Japanese Industrial Standard

Within an SQL statement, enclose dates in apostrophes.

Examples:

'2005-01-15'
'01/15/2005'
'15.01.2005'

Within an SQL statement, all date formats (USA, JIS, EUR, ISO) are accepted.

Time Data Type

Data Type	Comment	
TIME	Display / Input Formats	
HHMMSS	USA	hh:mm AM or PM
	EUR	hh.mm.ss
	ISO	hh.mm.ss
	JIS	hh:mm:ss

Figure 1-22 Time Data Type

Notes:

USA = US Standard
EUR = European Standard
ISO = International Standards Organization (Default)
JIS = Japanese Industrial Standard

Within an SQL statement, enclose times in apostrophes.

Examples:

'21.45.00'
'9:45 PM'
'21:45:00'

Graphic String Formats

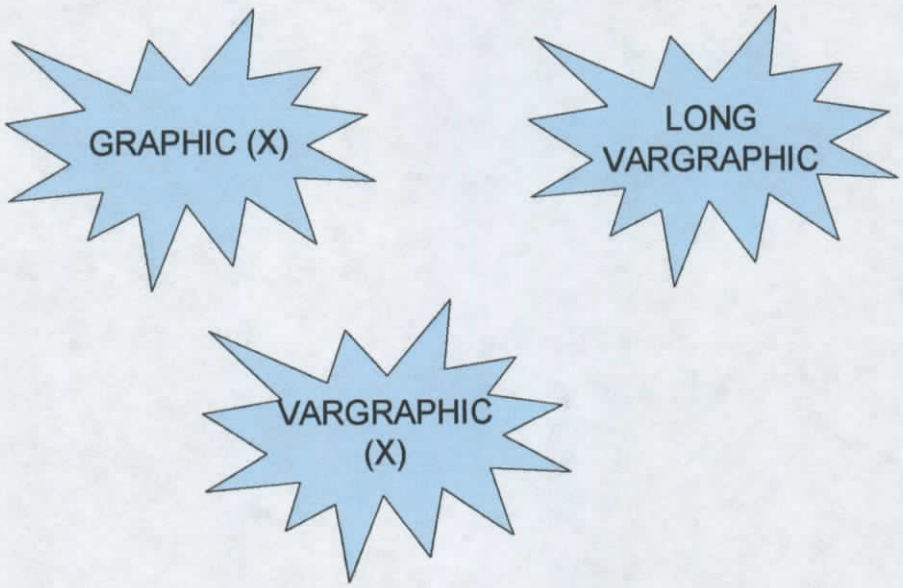


Figure 1-23 Graphic String Formats

Notes:

Graphic data types are analogous to the CHAR data types except in the graphic data types it takes two bytes to represent each character being stored (the alternate name for these data types is 'double byte character sets' (DBCS)).

Large Objects – The Need

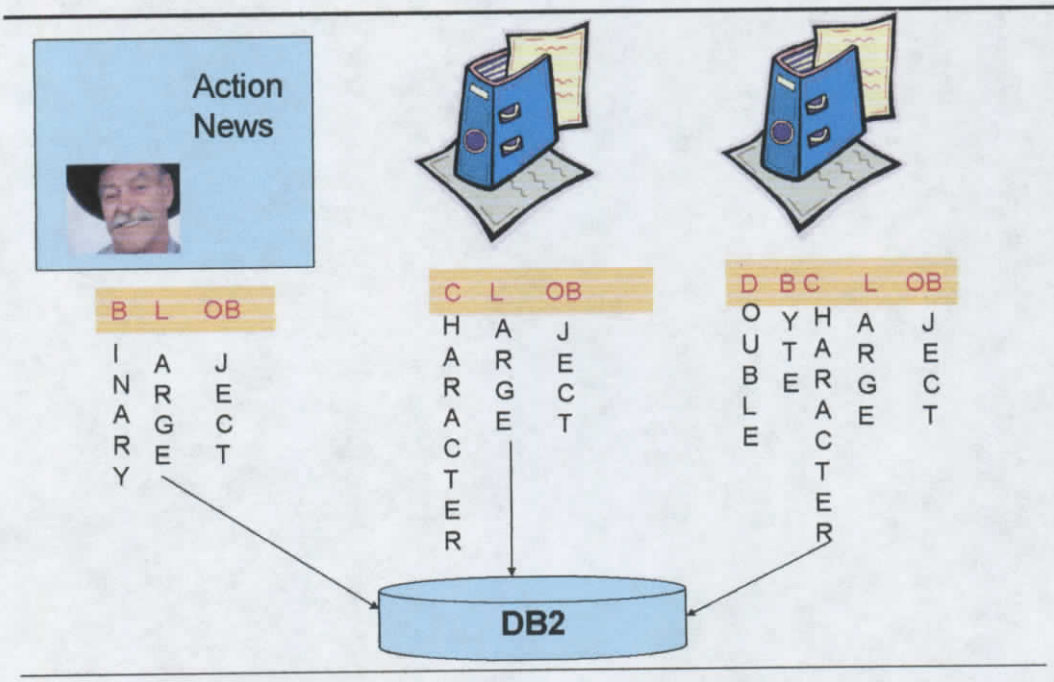


Figure 1-24 Large Objects – The Need

Notes:

- BLOBs can store images and sound data.
- CLOBs can store large amounts of character data
- DBCLOBS can store large amounts of double byte character data.

Create Table

```
Connect to musicdb;
Create table artist
(artno          smallint not null,
Name            varchar(50) with default 'abc',
Classification  char(1) not null,
Bio             clob(100k) logged,
Picture         blobl(2M) not logged compact)
```

Figure 1-25 Create Table

Notes:

These data types are specified just like the other data we've learned about.

This is an example of using these data types with DB2 UDB for UNIX, Windows, and OS/2.

DB2 Relational Extenders

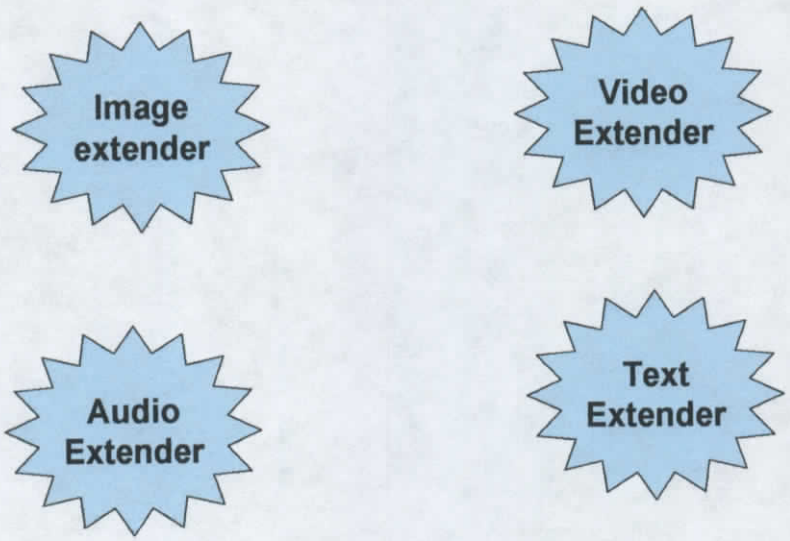


Figure 1-26 DB2 Relational Extenders

Notes:

The DB2 UDB products provide toolkits, called Extenders, that contain sets of predefined datatypes and functions. Each extender helps you to manage a particular kind of data, such as text, image, audio, or video.

NULL Characteristic

- **NOT NULL**
  - Column must always have a value

NOT NULL WITH DEFAULT

- Column must always have a value if we don't supply one, DB2 UDB does

- **"nullable"**
  - Column can be marked as having an 'unknown value'

Figure 1-27 NULL Characteristic

Notes:

A null characteristic tells DB2 UDB whether to allow a column to be marked as "The value for this column is unknown".

From an Editor that Talks to DB2 UDB

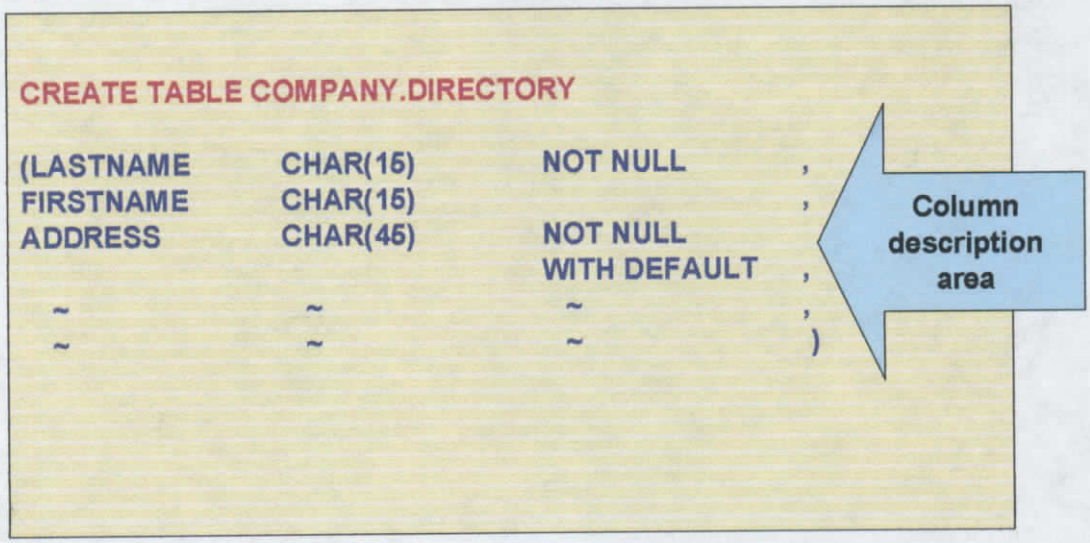


Figure 1-28 The language of Relational Database Management System

Notes:

This CREATE statement tells DB2 UDB that:

- The owner of the table is to be COMPANY.
- The simple name of the table is to be DIRECTORY.
- The name of the first column will be LASTNAME.
- Only character data is to be allowed in the first column.
- All values in the first column will be stored with 15 characters.
- The first column must always have a value.
- The name of the second column will be FIRSTNAME.
- Only character data is to be allowed in the second column.
- If a first name is not available for a row DB2 UDB is to mark it null (unknown).
- The name of the third column will be address.
- Only character data is to be allowed in the third column.
- If an address is not available for a given row, DB2 UDB is to fill the ADDRESS column with spaces.

Unique Key, Primary Key

- **Types**
  - SIMPLE – 1 COLUMN
  - COMPOUND – MULTIPLE COLUMNS
- **Characteristic**
  - Contains Unique Values
  - A table may have more than unique key
- **Primary Key – One per table**
  - All columns must be NOT NULL

Figure 1-29 Unique Key, Primary Key

Notes:

A table can have any number of unique keys, but only one defined primary key.

CREATE TABLE Statement

```
CREATE TABLE COUNTRY
( COUNTRY_CODE      CHAR (6)      NOT NULL      ,
  COUNTRY_NAME      CHAR (30)     NOT NULL      ,
  COL3_NAME         ~             ~             ,
  COL4_NAME         ~             ~             ,

  PRIMARY KEY (CONTRY_CODE)                     )
```

Figure 1-30 CREATE TABLE Statement

Notes:

Primary keys can be defined in the column description area of a CREATE TABLE statement.

Foreign Key

A Column or set of columns that contains values from some table's unique keys

Figure 1-31 Foreign Key

Notes:

Foreign keys are designed into our tables to define relationships between rows.

Example CREATE TABLE Statement

```
CREATE TABLE CITIES

(CITY_ID          CHAR(6)          NOT NULL          ,
CITY_NAME         CHAR(20)         NOT NULL          ,
COUNTRY_CODE      CHAR(6)          NOT NULL          ,
MILES_TO_LA       SMALLINT         NOT NULL          ,
POPULATION        INT                                ,
HEMISPHERE        CHAR(1)                            ,
AVG_INCOME        DECIMAL(9,2)                       ,
LAST_UPDATED      TIMESTAMP        NOT NULL
                                   WITH DEFAULT      ,

CITY_DESCRIPT     VARCHAR(300)                       ,
PRIMARY KEY       (CITY_ID)                          ,
FOREING KEY       (COUNTRY_CODE)   REFERENCES
                  COUNTRY                            )
```

Figure 1-32 Example CREATE TABLE Statement

Notes:

Foreign keys can be defined in the column description area of a CREATE TABLE statement.

In the table Cities, we are telling DB2 UDB that the COUNTRY_CODE column is a foreign key and that it contains values from the primary key of the COUNTRY table.

Primary Key, Foreign Key

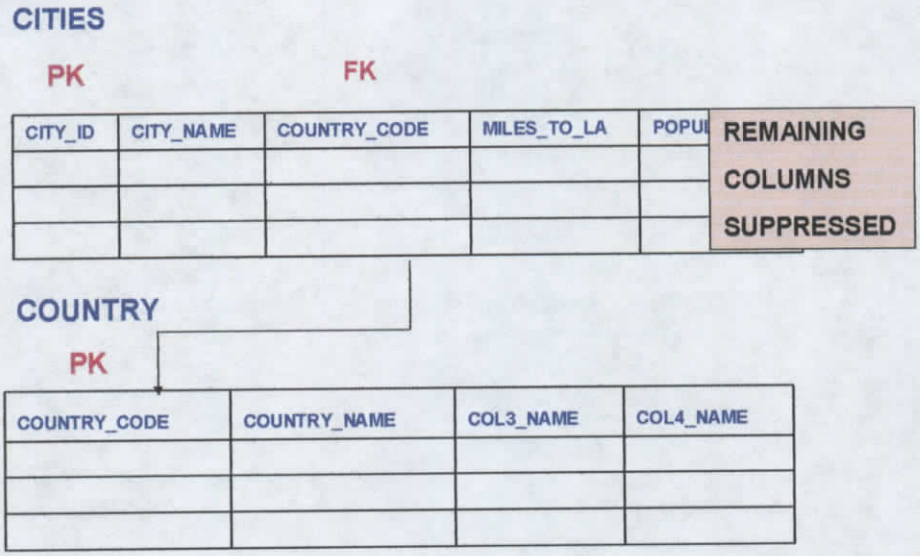


Figure 1-33 Primary Key, Foreign Key

Notes:

A city is related to its country by the common values in the COUNTRY_CODE columns. Should we need to produce a report with information about a city and the country in which it is located, we can find a city's country by searching for a matching COUNTRY_CODE.

The Rules DB2 Enforces to Maintain Referential Integrity



Figure 1-35 The Rules DB2 Enforces to Maintain Referential Integrity

Notes:

Primary key values cannot be null.

A foreign key contains a set of values from some table's unique key.

Adding Primary and Foreign Keys to an Existing table

When altering the table ...

```
ALTER TABLE BENEFIT_CLAIMS
  ADD PRIMARY KEY (CALIM_NO)
    FOREIGN KEY (GOV_ID)
    REFERENCES EMPL (GOV_ID)
```

Figure 1-36 Adding Primary and Foreign Keys to an Existing table

Notes:

Referential integrity can be added after the tables have been created, via an ALTER TABLE statement.

Processing Rules Enforced to Maintain Referential Integrity

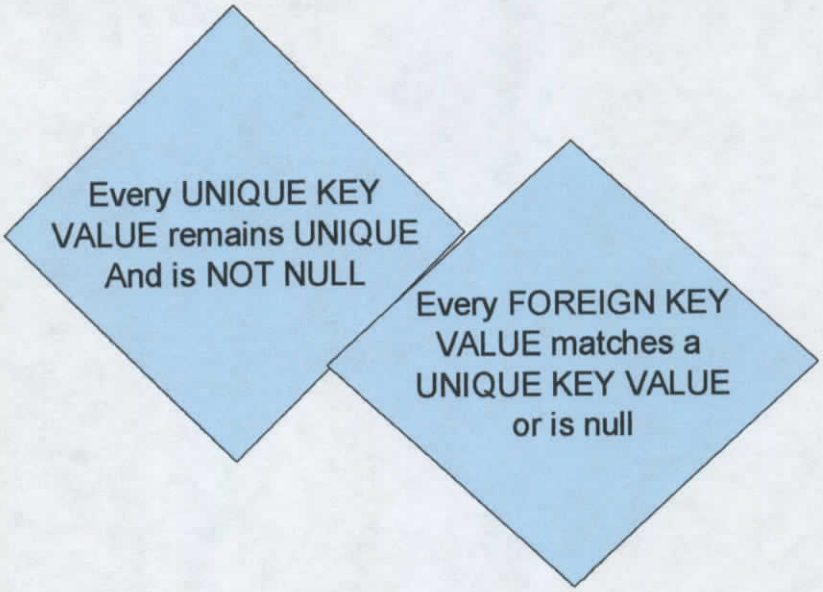


Figure 1-37 Processing Rules Enforced to Maintain Referential Integrity

Notes:

Defined referential integrity causes DB2 to enforce these rules.

Primary Key Violation

EMPNO	LASTNAME	FIRSTNAME	DEP	GOVT_ID	SALARY
000010	HAAS	CHRISTINE	RED	888-882794	52750.00
000020	THOMPSON	MICHAEL	BLUE	888-89-4261	31000.00
000030	KWAN	SALLY	GRE	888-88-9456	33000.00

DEP	DEPTNAME	MANAGER	DIVISION
BLU	PLANNING	000020	EASTERN
GRE	OPERATION	000080	WESTERN
RED	MARKETING	000010	EASTERN

New employee –INSERT FAILS
| 000010 |DEAN | ANN |BLU | 888-32-3265 | 33250.00

Figure 1-38 Primary Key Violation

Notes:

Why did the insert fail?

Foreign Key Violation

EMPNO	LASTNAME	FIRSTNAME	DEP	GOVT_ID	SALARY
000010	HAAS	CHRISTINE	RED	888-882794	52750.00
000020	THOMPSON	MICHAEL	BLUE	888-89-4261	31000.00
000030	KWAN	SALLY	GRE	888-88-9456	33000.00

DEP	DEPTNAME	MANAGER	DIVISION
BLU	PLANNING	000020	EASTERN
GRE	OPERATION	000080	WESTERN
RED	MARKETING	000010	EASTERN

New employee –INSERT FAILS
| 000110 |DEAN | ANN |BRN | 888-32-3265 | 33250.00

Figure 1-39 Foreign Key Violation

Notes:

Why did the insert fail?

Potential SYSADM Task

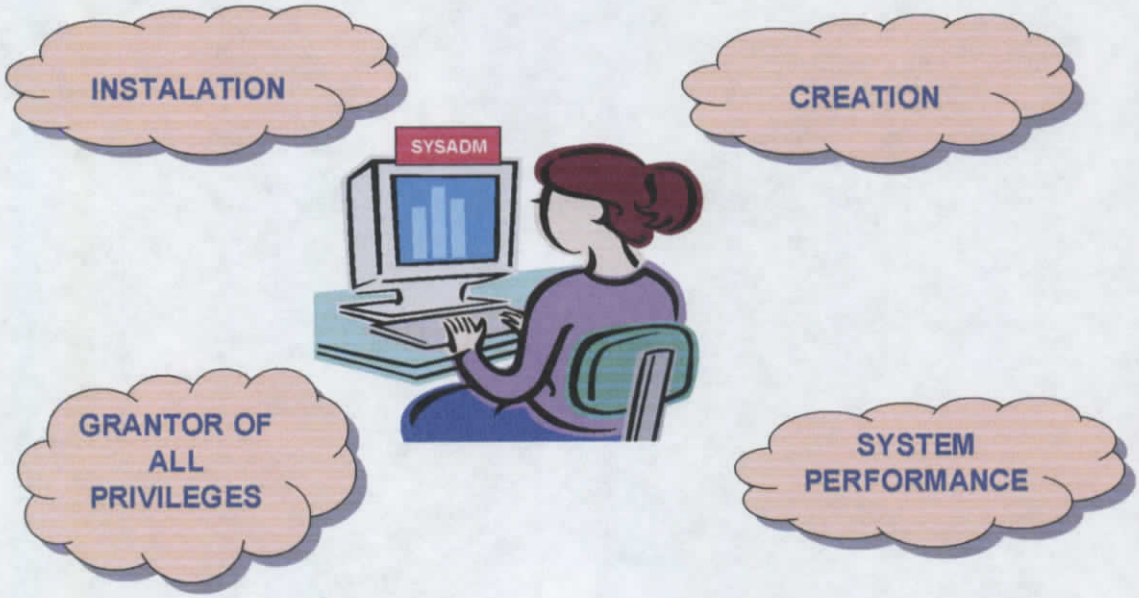


Figure 1-40 Potential SYSADM Task

Notes:

- System programmer and/or the SYSADM does the physical install.
- The SYSADM creates the system level objects (database).
- The SYSADM grants other IDs appropriate privileges (DBADM, CREATETAB, and so forth.
- SYSADM monitors the overall DB2 UDB system performance.

Potential DBADM Task

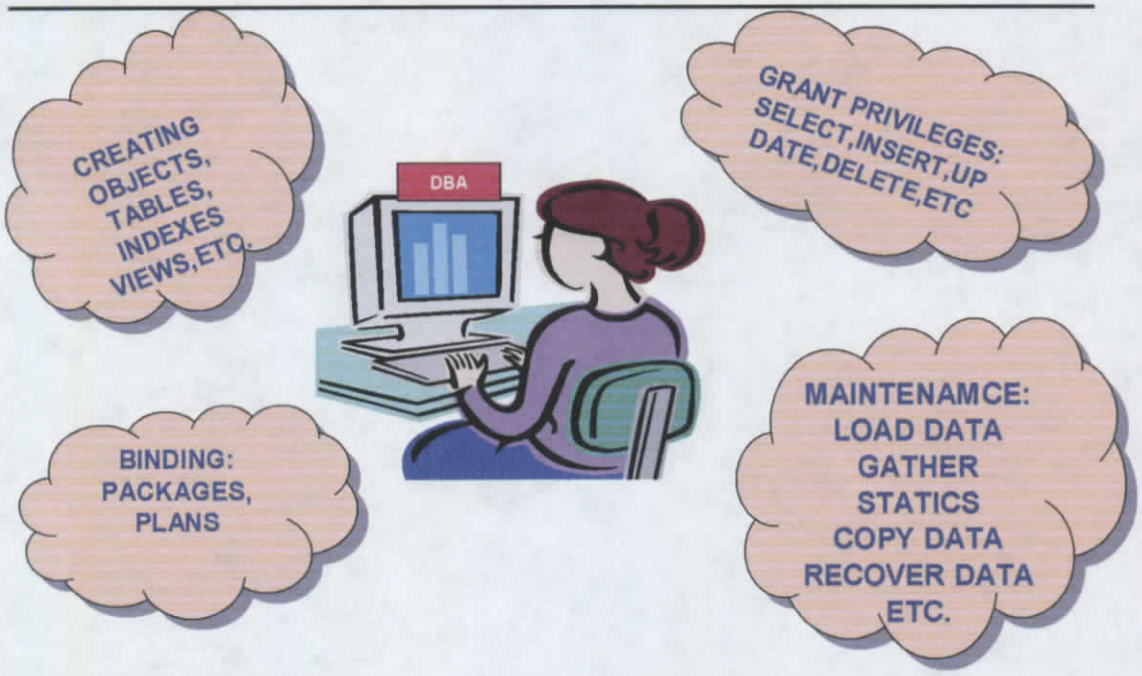


Figure 1-41 Potential DBADM Task

Notes:

- Responsible for building and maintaining the objects within the database.
- Grants to other IDs privileges to access and/or changed objects within the database.
- In many shops the Database Administrator (DBA – the person(s) with DBADM authority) perform the production binds (prepare application programs to work with DB2 UDB).
- DBADM is responsible for the running of most DB2 UDB utilities.

Potential Programmer Task

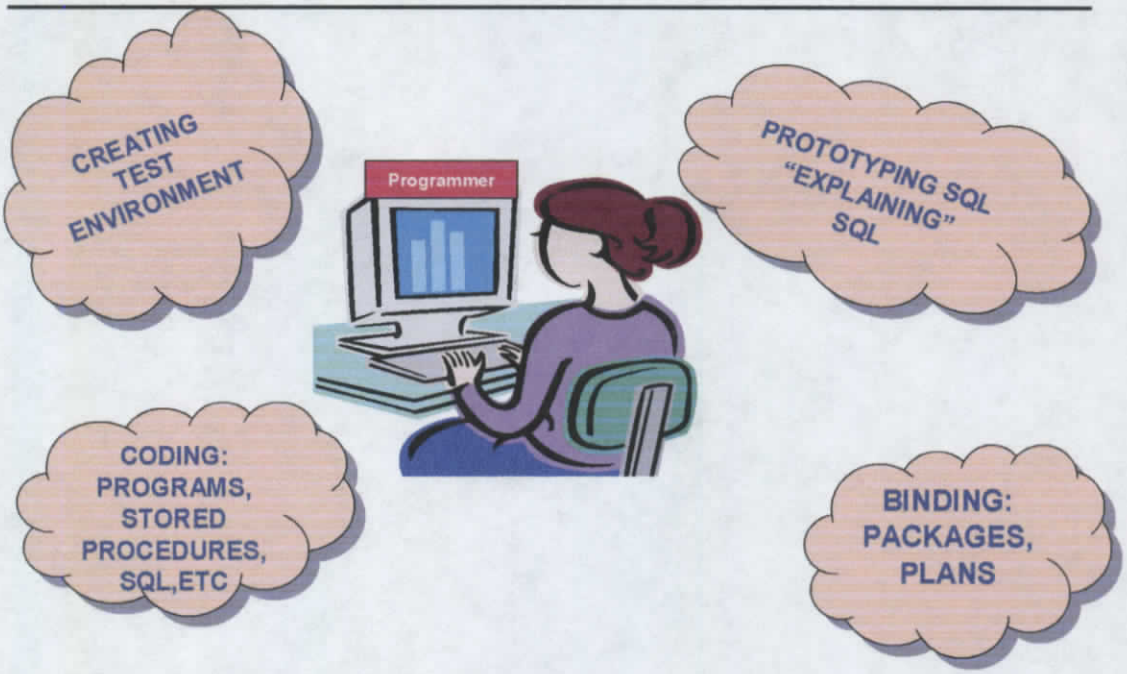


Figure 1-42 Potential Programmer Task

Notes:

- Programmers may be involved with creating test environments for new applications.
- Programmers prototype SQL (write and test the SQL before it is used in production).
- Programmers are concerned with the efficiency of their SQL. Explain is a DB2 UDB service that 'tells' us how DB2 UDB is going to execute our SQL.
- Programmer physically embed SQL in host language programs.
- Programmers prepare (precompile and bind) their programs for execution.

Potential End – User Tasks

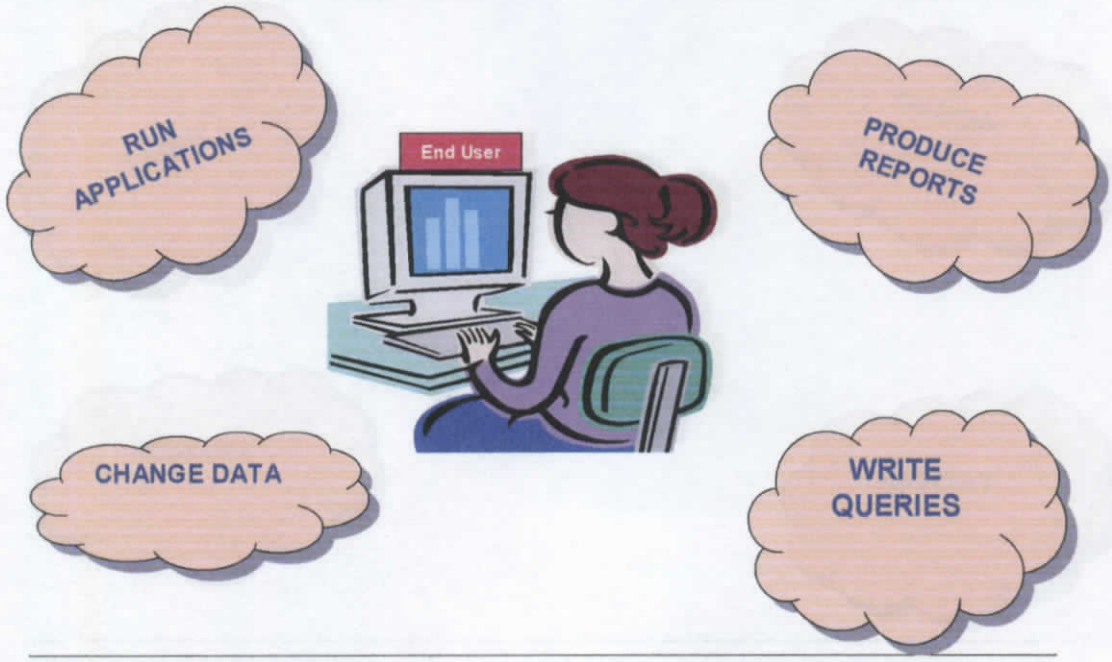


Figure 1-43 Potential End – User Tasks

Notes:

Anyone executing an SQL statement is an end-user.

- SQL can be in applications end-users run.
- The SQL within the applications may be changing data, adding data, removing data or reading data.
- The end-user may construct (write) their own queries that change data, add data, remove data, or read data.

Unit Summary

Since completing this unit, you should be able to :

- List three services provided by a database manager
- Define a RELATION
- Name the language used 'talk' to a relational database manager (RDBM)
- List three characteristics assigned to each column
- List several data types that can be assigned to a column
- List three forms of Key used in DB2 UDB tables

Figure 1-45 Unit Summary

Notes:

- A table format implies columns and rows.
- Relational database managers understand SQL.
- Every column has a column name, a data type assigned to it, and a null characteristic.
- The kissing services are Concurrency, Integrity, and Security.
- Integer, smallint, and decimal are numeric data types. Char and varchar are character data types. Date, time, and timestamp are special data types designed to store dates, times, and timestamp values.
- Unique key is a column or set of columns whose values are unique. A table can have any number of unique keys.
- A primary key is one of the unique keys that is defined to the relational database manager. A table can have only one defined primary key.
- A foreign key is a column or set of columns that contain values from some table's unique key.

Checkpoint

Exercise — Unit 1 Checkpoint

1. What are the "kissing" (CIS) services a database manager provides?

   _____

   _____

   _____

2. What is the language that DB2 UDB understands?

   _____

3. List three characteristic assigned to every column.

   _____

   _____

   _____

4. List the three null characteristics.

   _____

   _____

   _____

5. List several of the DB2 UDB column data types.

   _____

   _____

   _____

   _____

6. What term is interchangeable with the term RELATION

   _____

7. List the three forms of KEY discussed in this unit.

_____

_____

_____

8. Create table TEST_EMP with the following columns, add PRIMARY KEY:

▪ EMPNO	CHAR(6)	NOT NULL
▪ LASTNAME	VARCHAR(15)	NOT NULL
▪ WORKDEPT	CHAR(3)	
▪ HIREDATE	DATE	
▪ SALARY	DECIMAL(9,2)	
▪ BONUS	DECIMAL(9,2)	

Unit 2 . How Does An End-User Use DB2 UDB?

What This Unit Is About

This unit explains some of the tasks an End-User may perform while working with DB2 UDB.

What You Should Be Able to Do

After completing this unit, you should be able to:

- List several ways to 'talk' to DB2 UDB.
- List multiple ways to generate an SQL statement.
- List several ways to ADD, REMOVE, or CHANGE table rows.
- List several ways to READ data and produce report.

How You Will Check Your Progress

Accountability:

- Checkpoint questions

Unit Objectives

After completing this unit, you should be able to :

- List several ways to talk to DB2
- List multiple ways to generate an SQL statement
- List several ways to ADD, REMOVE, or CHANGE table rows
- List several ways to READ data and produce reports

Figure 2-1 Unit Objectives

Notes:

2.1 Ways to Run SQL

How Do I ' Talk' to DB2 UDB?

- **DB2 UDB UNIX/Intel**
  - Command Line Processor (CLP)
  - Command Center
- **DB2 UDB OS/390**
  - SPUFI
  - QMF
- **Other graphical tools (for example, QMF for Windows)**

Figure 2-2 How Do I ' Talk' to DB2 UDB?

Notes:

Click the appropriate icon to get to the CLP, Command Center, or QMF for Windows. In OS/390, SPUFI and QMF are usually reached from the ISPF menu.

Command Line Processor - Interactive

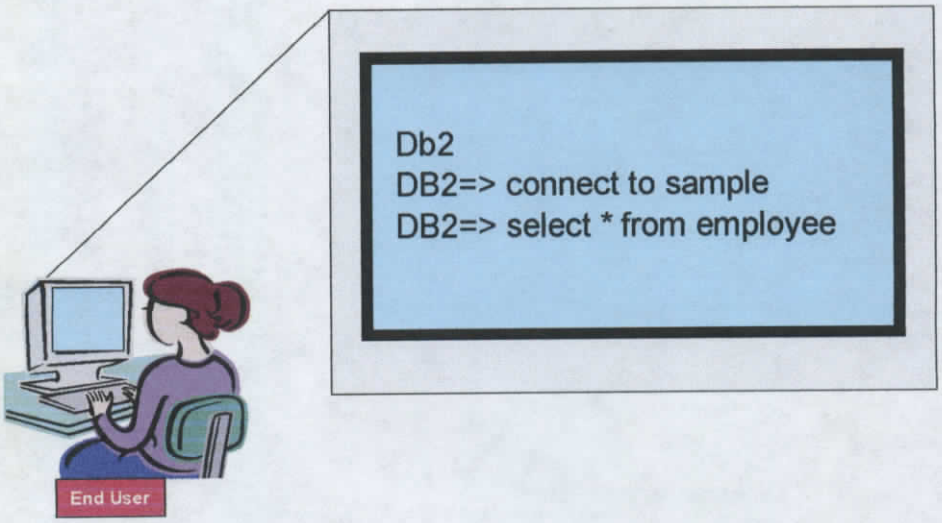


Figure 2-3 Command Line Processor - Interactive

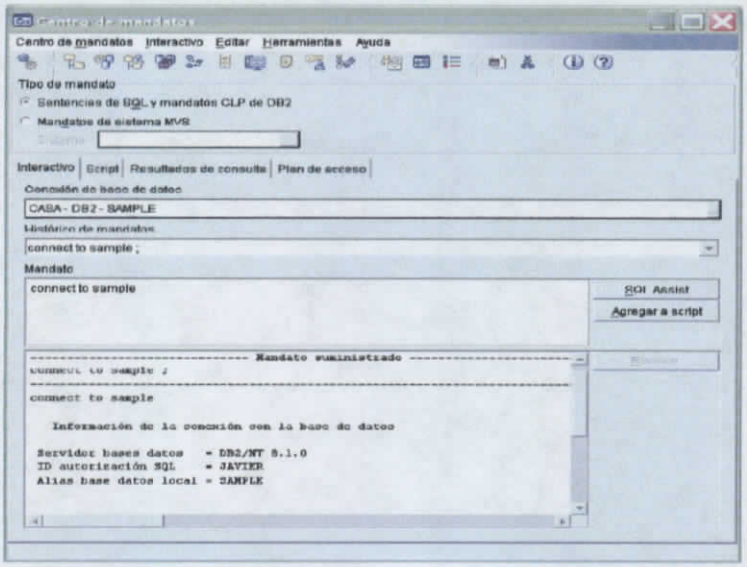
Notes:

Command Line Processor will also accept SQL statements in an non-interactive mode, by prefixing the SQL statement with the term DB2.

Instead of typing DB2 and pressing enter, which evokes the interactive mode of CLP, star each SQL statement with DB2, as in the following example:

```
db2 connect to database_a
db2 select lastname,phone from france.paris where firstname = 'PAT'
```

Command Center



- Graphical Command Line Processor

Figure 2-4 Command Center

Notes:

'Connect' to the DB2 UDB that will execute the SQL.

SQL statement, DB2 UDB commands, and operating system commands can be run from the Command Center.

A stack of SQL statements can be executed from the Scrip tab.

2.2 Writing SQL

Who is a End – User?

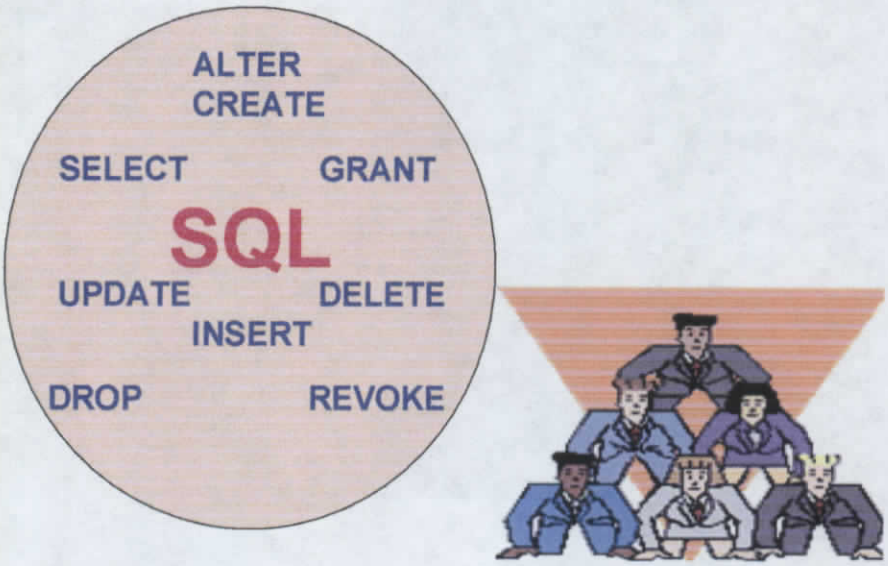


Figure 2-5 Who is a End – User?

Notes:

End-user is anyone that through their actions, issues an SQL statement.

Running Programs that 'Talk' to DB2 UDB

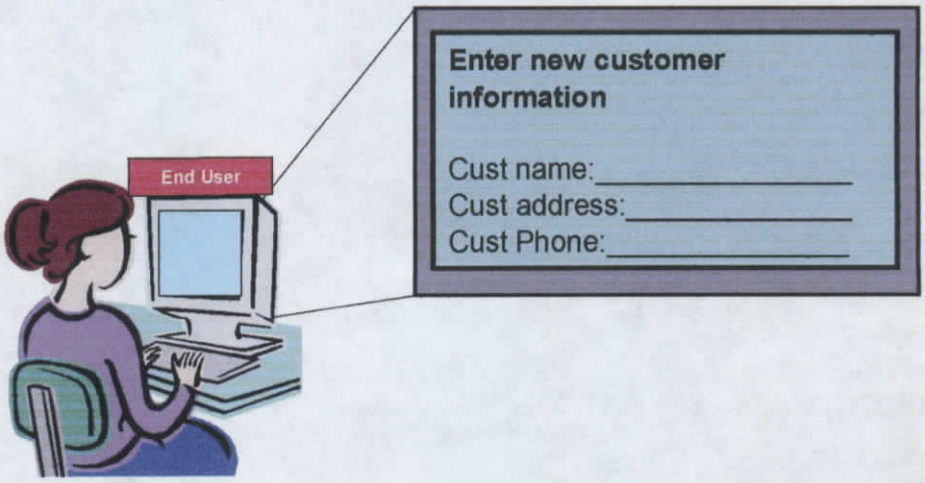


Figure 2-6 Running Programs that 'Talk' to DB2 UDB

Notes:

Programs can contain SQL statement. People running those programs are end-users.

And if I'm Writing my SQL?

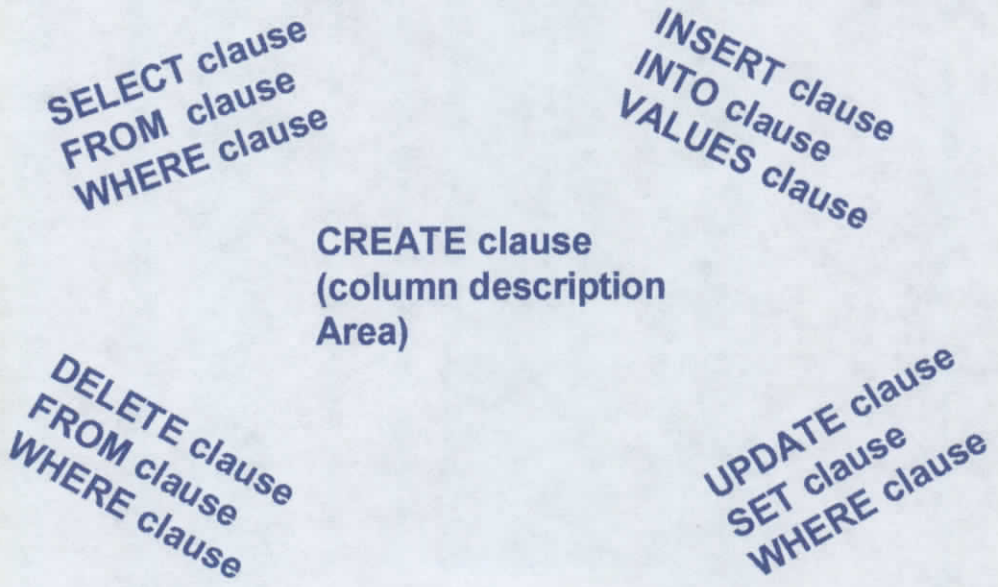


Figure 2-7 And if I'm Writing my SQL?

Notes:

SQL statements are made up of clauses. The sequence in which the clauses are written is the structure of the language.

Four SELECT Statement Clauses

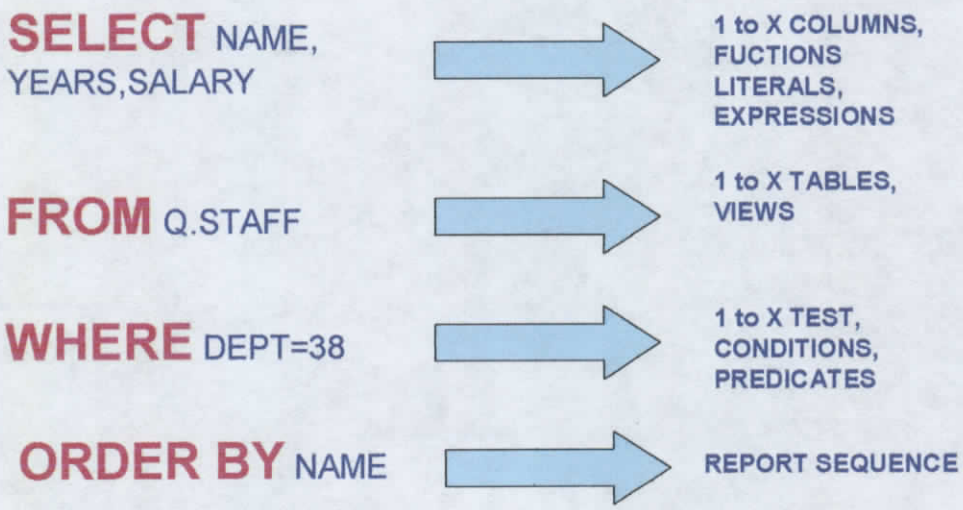


Figure 2-8 Four SELECT Statement Clauses

Notes:

Every SELECT must have a SELECT and FROM clause. The remaining clauses are only used when we need their services.

The SELECT clause tells DB2 UDB which column(s) of information you want on your report.

The FROM clause specifies which table(s) contain the requested data.

The WHERE clause tells DB2 UDB from which row(s) the SELECT clause information is to be retrieved; such as, information about department 38.

The ORDER BY clause specifies the order of the information on the report.

The sequence of the clauses must be coded as shown here: SELECT followed by FROM, WHERE, then ORDER BY.

To Retrieve All Columns for All Rows

```
SQL QUERY                                        MODIFIED LINE 1

SELECT *                -- ASTERISK SPECIFIES ALL COLUMNS
FROM STAFF
                        -- NO WHERE CLAUSE MEANS RETURN ALL ROWS




*** END ***
```

Figure 2-9 To Retrieve All Columns for All Rows

Notes:

An asterisk in a SELECT clause means: return all columns from this table. This could also be written: SELECT Q.STAFF.*

FROM clause is required in every query.

The absence of a WHERE clause tells DB2 UDB: We want data from every row in the table.

Two hyphens, side by side, denote a comment.

To SELECT Specified Rows

```
SELECT JOB,NAME
FROM STAFF
WHERE DEPT=20
```

JOB	NAME
MGR	SANDERS
SALES	PERNAL
CLERK	JAMES
CLERK	SNEIDER

Figure 2-10 To SELECT Specified Rows

Notes:

This query returns two columns of data from the set of rows whose DEPT value is 20.

Base Table

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM	
10	**Sanders**	**20**	**Mgr**	**7**	**18357.50**		
20	**Pernal**	**20**	**Sales**	**8**	**18171.25**	**612.45**	
30	Marenghi	38	Mgr	5	17506.75		
40	O'Brien	38	Sales	6	18006.00	846.55	
50	Hanes	15	Mgr	10	20659.80		
60	Quigley	38	Sales		16808.30	650.25	
70	Rothman	15	Sales	7	16502.83	1152.00	
80	**James**	**20**	**Clerk**		**13504.60**	**128.20**	
90	Koonitz	42	Sales	6	18001.75	1386.70	
100	Plotz	42	Mgr	7	18352.80		
110	Ngan	15	Clerk	5	12508.20	206.60	
.	.	.	.	.	.	.	
180	Abrahams		38	Clerk	3	12009.75	236.50
190	**Sneider**	**20**	**Clerk**	**8**	**14252.75**	**126.50**	
200	Scoutten	42	Clerk		11508.60	84.20	
.	.	.	.	.	.	.	

Figure 2-11 Base Table

Notes:

The WHERE clause identifies the set of rows from which the data is to be returned.

Conceptual Intermediate Table

10	Sanders	20	Mgr	7	18357.50	
20	Pernal	20	Sales	8	18171.25	612.45
80	James	20	Clerk		13504.60	128.20
190	Sneider	20	Clerk	8	14252.75	126.50

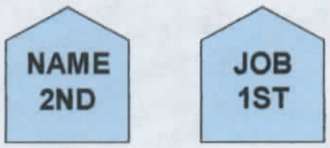


Figure 2-12 Conceptual Intermediate Table

Notes:

DB2 UDB applies the SELECT clause against the conceptual intermediate table retrieving the JOB and NAME values (in that left-to-right sequence) from every row in the intermediate table.

Numeric Comparisons

TO QUALIFY ROWS BY NUMERIC DATA

```
WHERE DEPT = 20          --(integer)
AND COMM = 612.45        --(decimal pt.)
AND (SALARY=18171.25     --(no comma)
OR SALARY = 18000)       --(same as 18000.00)
```

Figure 2-13 Numeric Comparisons

Notes:

Numeric data is either decimal expressed as DECIMAL(x,y), integer (SMALLINT or INTEGER), or floating-point (FLOAT,FLOAT(x), REAL, or DOUBLE PRECISION).

SQL Comparison Alternatives

OPERATORS	(>,<>,<,>=,<=)
BOOLEAN	(AND,OR)
PARTIAL VALUES	(LIKE '_A%')
VALUE IN ROW	(COMM>SALARY)
CALCULATED VALUE	(COMM/12)

```
SELECT * FROM EMPLOYEE
WHERE ((SALARY + COMM) > 40000)
      AND WORKDEPT LIKE 'B%'
```

Figure 2-14 SQL Comparison Alternatives

Notes:

There are two ways to write 'not equal' in SQL OS/390: ¬ =, and <>.

In some environments != is accepted to mean 'not equal'.

Partial value searches using the LIKE keyword can only be performed on character data.

Functions – Two Types

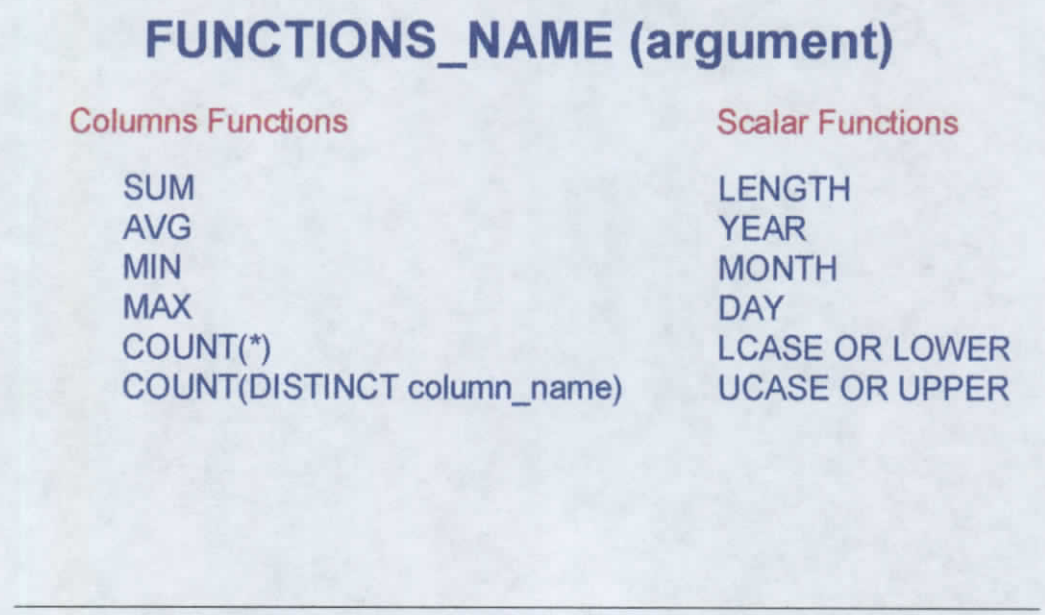


Figure 2-15 Functions – Two Types

Notes:

- Column functions produce one summary row for each 'set' of rows.
- Scalar functions produce one value for each row within the set.
- The above column functions are the most commonly used column functions. Each release of DB2 UDB provides additional functions.

Column Functions

```
SELECT SUM(SALARY) AS SUM_SALARY,
       COUNT(*) AS COUNT
FROM   STAFF
WHERE DEPT = 20
```

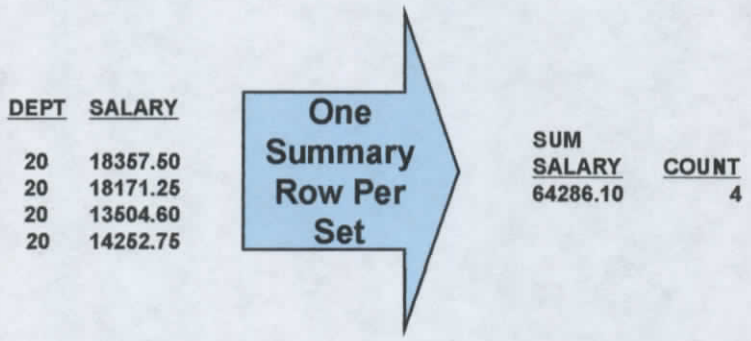


Figure 2-16 Column Functions

Notes:

COUNT(*) counts rows that meet the WHERE clause conditions. SUM(SALARY) adds up the known values in the salary column for the set of rows that have a DEPT value of 20.

Scalar Functions

```
SELECT DIGITS(DEPTNUMB) AS C_DEPT
FROM ORG
```

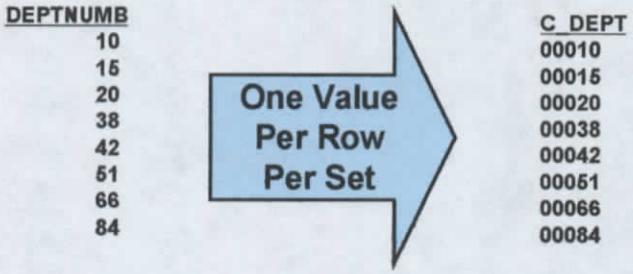


Figure 2-17 Scalar Functions

Notes:

Some additional scalar functions are: MONTH, DAY, DAYS, COALESCE, SUBSTR, DATE, DECIMAL, DIGITS, ROUND, TRIM, NULLIF, to name a few.

UPPER

```
SELECT ID,NAME,DEPT,JOB,SALARY
FROM STAFF
WHERE UPPER(JOB) = 'MGR'
```

ID	NAME	DEPT	JOB	SALARY
10	Sanders	20	Mgr	18357.50
30	Marenghi	38	Mgr	17506.75
50	Hanes	15	Mgr	20659.80
100	Plotz	42	Mgr	18352.80
140	Fraye	51	Mgr	21150.00
160	Molinare	10	Mgr	22959.20
210	Lu	10	Mgr	20010.00
240	Daniels	10	Mgr	19260.25
260	Jones	10	Mgr	21234.00
270	Lea	66	Mgr	18555.50
290	Quill	84	Mgr	19818.00

Figure 2-18 UPPER

Notes:

Many time data is stored with mixed case characters either on purpose or because of data entry errors. The UPPER scalar function allows us to perform case insensitive searching.

The function may also be evoked under the name UCASE().

Check the reference manual for you environment to see if UPPER() is supported in your DB2 UDB release.

Conceptual Execution of a SELECT

BASE TABLE

JOB	SAL	DEP
S	31000	BLU
M	32000	RED
S	30000	BLU
C	27000	GRE
S	33000	GRE
M	31000	BLU
S	32000	RED
C	28000	GRE
S	30000	RED
M	33000	GRE
S	31000	RED
S	35000	GRE
C	27000	BLU
S	29000	RED
S	29000	BLU

WHERE

JOB	SAL	DEP
S	31000	BLU
S	30000	BLU
C	27000	GRE
S	33000	GRE
S	32000	RED
C	28000	GRE
S	30000	RED
S	31000	RED
S	35000	GRE
C	27000	BLU
S	29000	RED
S	29000	BLU

GROUP BY

JOB	SAL	DEP
C	27000	BLU
S	31000	BLU
S	29000	BLU
S	30000	BLU
C	27000	GRE
C	28000	GRE
S	33000	GRE
S	35000	GRE
S	32000	RED
S	30000	RED
S	31000	RED
S	29000	RED

Figure 2-20 Conceptual Execution of a SELECT

Notes:

- Conceptually, the WHERE clause pulls a copy of all qualifying rows into an 'intermediate' table.
- The GROUP BY clause sorts the intermediate table into sets based on the values in the grouping columns.
  - One group for each unique set of values within the grouping columns.
- The HAVING clause qualifies groups for further processing.

Conceptual Execution of a SELECT (Cont)

HAVING

JOB	SAL	DEP
S	31000	BLU
S	29000	BLU
S	30000	BLU
S	33000	GRE
S	35000	GRE
S	32000	RED
S	30000	RED
S	31000	RED
S	29000	RED

SELECT

DEP	JOB	AVG(SAL)
BLU	S	30000
GRE	S	34000
RED	S	30500

ORDER BY

DEP	JOB	AVG(SAL)
GRE	S	34000
RED	S	30500
BLU	S	30000

Figure 2-21 Conceptual Execution of a SELECT (Cont)

Notes:

- The HAVING clause has qualifies groups for further processing.
- The SELECT list is applied to the qualified groups creating one summary row (result row) for each group.
- The ORDER BY sorts the result rows based on values in the ORDER BY columns.

SELECT Clause – Column Naming

In a SELECT statement

```
SELECT something AS  new_name,
       expression AS "multi part name" ...

SELECT DEP AS DEPARTMENT,
       SAL + COMM AS EARNINGS

FROM ...
```

DEPARTMENT	EARNINGS
BLU	64000
RED	58000
.	.

Figure 2-22 SELECT Clause – Column Naming

Notes:

The AS keyword is optional.

NOTE: forgetting a comma between two columns within a select list may or may not cause a syntax error. It may cause DB2 UDB to extract values from the first column but apply the name of the second column to the result.

Inserting Rows into the Table

```
INSERT INTO TESTEMP
VALUES('0000111','SMITH','C01','1998-06-25',25000,NULL),
      ('0000113','JONES','C01','2001-06-25',25000,NULL),
      ('0000114','THOMPSON','C01','2001-06-25',25000,NULL)

OR

INSERT INTO TESTEMP
(EMPNO,LASTNAME,WORKDEPT,HIREDATE,SALARY)
VALUES('0000111','SMITH','C01','1998-06-25',25000,NULL),
      ('0000113','JONES','C01','2001-06-25',25000,NULL),
      ('0000114','THOMPSON','C01','2001-06-25',25000,NULL)
```

Figure 2-23 Inserting Rows into the Table

Notes:

Use an INSERT statement to add rows into a table. Inserting a row via a view inserts the row into the table on which the view is based. Therefore, inserts via a view are only possible if the view contains all columns all of the NOT NULL columns in the base table.

When using the syntax of the first example a value for each of the table's columns must be provided and, those values must be in the sequence of the columns into which they must go. Each set of parenthesis in the first example is adding a row to the table. Prior to DB2 UDB V7, only one row could be added per insert statement.

In the second example, a column list is specified. The number of values provided for a row via the VALUES clause must be equal to the number of column names in the column list. The first value is inserted in the first column of the list, the second value in the second column, and so on.

Early releases of DB2 UDB supported inserting only one row per INSERT statement. Check the SQL Reference manual for your environment to ensure support of multi-row insert.

Update Columns

Before:

EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
0000111	*SMITH*	*C01*	*1998-06-25*	*25000*	-
0000113	JONES	C01	2001-06-25	25000	-
0000114	THOMPSON	C01	2001-06-25	25000	-
000010	HAAS	A00	1965-01-1	52750.00	1000.00
000020	THOMPSON	B01	1973-10-10	41250.00	800.00
000030	KWAN	C01	1975-04-05	38250.00	800.00

```
UPDATE TESTEMP
SET BONUS = 500,
    SALARY = 26000
WHERE EMPNO='000111'
```

After:

EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
0000111	*SMITH*	*C01*	*1998-06-25*	*26000*	*500*
0000113	JONES	C01	2001-06-25	25000	-
0000114	THOMPSON	C01	2001-06-25	25000	-
000010	HAAS	A00	1965-01-1	52750.00	1000.00
000020	THOMPSON	B01	1973-10-10	41250.00	800.00
000030	KWAN	C01	1975-04-05	38250.00	800.00

Figure 2-24 Update Columns

Notes:

The UPDATE statement changed the values of the specified columns for one or more rows of a table or view.

Our new colleague, Mr. Smith (employee number 000111) receives a first bonus and a salary raise. Therefore, we must set BONUS to 500 and SALARY to 2600

If more than one column is changed, the columns must be separated by commas. The SET keyword must precede the first column and must not be repeated.

To set BONUS to NULL, the UPDATE statement must be coded as follows:

```
UPDATE TESTEMP
  SET BONUS = NULL
WHERE EMPNO = '000111'
```

Delete Rows

Before:

EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
0000111	*SMITH*	*C01*	*1998-06-25*	*25000*	-
0000113	JONES	C01	2001-06-25	25000	-
0000114	THOMPSON	C01	2001-06-25	25000	-
000010	HAAS	A00	1965-01-1	52750.00	1000.00
000020	THOMPSON	B01	1973-10-10	41250.00	800.00
000030	KWAN	C01	1975-04-05	38250.00	800.00

```
DELETE FROM TESTEMP
WHERE EMPNO = '000111'
```



After:

EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
0000113	JONES	C01	2001-06-25	25000	-
0000114	THOMPSON	C01	2001-06-25	25000	-
000010	HAAS	A00	1965-01-1	52750.00	1000.00
000020	THOMPSON	B01	1973-10-10	41250.00	800.00
000030	KWAN	C01	1975-04-05	38250.00	800.00

Figure 2-25 Delete Rows

Notes:

Mr. Smith left the company. Therefore, we delete the row with employee number 000111.

The DELETE statement removes entire rows from a table. Depending on the search condition of the WHERE clause, the DELETE statement removes zero, one, or more rows of a table.

The delete specific rows, use the WHERE clause to specify the rows to be deleted.

If you **omit the WHERE clause, all rows** of the table or view are **deleted.**

Delete Rules

When a row whose referenced unique key value is Deleted, what should be done with the referencing rows?

The table designer chooses one of the following DELETE RULES:

CASCADE

SET NULL

RESTRICT

Figure 2-26 Delete Rules

Notes:

There are three delete rules to choose from.

ON DELETE RESTRICT is the default.

Delete Rules – ON DELETE CASCADE

DEP	DEPTNAME	MANAGER	DIVISION
BLU	PLANNING	000020	EASTERN
GRE	OPERATION	000080	WESTERN
RED	MARKETING	000010	EASTERN

EMPNO	LASTNAME	FIRSTNAME	DEP	GOVT_ID	SALARY
000010	HAAS	CHRISTINE	RED	888-882794	52750.00
000020	THOMPSON	MICHAEL	BLUE	888-89-4261	31000.00
000030	KWAN	SALLY	GRE	888-88-9456	33000.00

DEP	DEPTNAME	MANAGER	DIVISION
GRE	OPERATION	000080	WESTERN
RED	MARKETING	000010	EASTERN

EMPNO	LASTNAME	FIRSTNAME	DEP	GOVT_ID	SALARY
000010	HAAS	CHRISTINE	RED	888-882794	52750.00
000030	KWAN	SALLY	GRE	888-88-9456	33000.00

Figure 2-27 Delete Rules – ON DELETE CASCADE

Notes:

In this example, the referenced table is the department table. The foreign key EMPNO.DEP was defined with an ON DELETE CASCADE rule.

What happens when the BLU DEP row is deleted from the department table with ON DELETE CASCADE as the delete rule? All employees in the BLU DEP are also deleted.

Delete Rules – ON DELETE SET NULL

DEP	DEPTNAME	MANAGER	DIVISION
BLU	PLANNING	000020	EASTERN
GRE	OPERATION	000080	WESTERN
RED	MARKETING	000010	EASTERN

EMPNO	LASTNAME	FIRSTNAME	DEP	GOVT_ID	SALARY
000010	HAAS	CHRISTINE	RED	888-882794	52750.00
000020	THOMPSON	MICHAEL	BLUE	888-89-4261	31000.00
000030	KWAN	SALLY	GRE	888-88-9456	33000.00

DEP	DEPTNAME	MANAGER	DIVISION
GRE	OPERATION	000080	WESTERN
RED	MARKETING	000010	EASTERN

Unknown

EMPNO	LASTNAME	FIRSTNAME	DEP	GOVT_ID	SALARY
000010	HAAS	CHRISTINE	RED	888-882794	52750.00
000020	THOMPSON	MICHAEL		888-89-4261	31000.00
000030	KWAN	SALLY	GRE	888-88-9456	33000.00

Figure 2-28 Delete Rules – ON DELETE SET NULL

Notes:

In this example, the referenced table is the department table. The foreign key EMPNO.DEP was defined with an ON DELETE SET NULL rule.

When the BLU DEP row is deleted from the department table, DB2 will set the DEP value for all BLU DEP employees, to NULL.

Delete Rules – ON DELETE RESTRICT

DEP	DEPTNAME	MANAGER	DIVISION
BLU	PLANNING	000020	EASTERN
GRE	OPERATION	000080	WESTERN
RED	MARKETING	000010	EASTERN

EMPNO	LASTNAME	FIRSTNAME	DEP	GOVT_ID	SALARY
000010	HAAS	CHRISTINE	RED	888-882794	52750.00
000020	THOMPSON	MICHAEL	BLUE	888-89-4261	31000.00
000030	KWAN	SALLY	GRE	888-88-9456	33000.00

DEP	DEPTNAME	MANAGER	DIVISION
BLU	PLANNING	000020	EASTERN
GRE	OPERATION	000080	WESTERN
RED	MARKETING	000010	EASTERN

EMPNO	LASTNAME	FIRSTNAME	DEP	GOVT_ID	SALARY
000010	HAAS	CHRISTINE	RED	888-882794	52750.00
000020	THOMPSON	MICHAEL	BLUE	888-89-4261	31000.00
000030	KWAN	SALLY	GRE	888-88-9456	33000.00

Figure 2-29 Delete Rules – ON DELETE RESTRICT

Notes:

In this example, the referenced table is the department table. The foreign key EMPNO.DEP was defined with an ON DELETE RESTRICT rule.

Before DB2 will allow the BLU DEP row to be deleted from the department table, it checks to ensure that there are no employee rows with BLU DEP values. If DB2 finds one employee row with a value of BLU in the DEP column, the delete fails (is restricted).

Unit Summary

Since completing this unit, you should be able to :

- List several ways to talk to DB2
- List multiple ways to generate an SQL statement
- List several ways to ADD, REMOVE, or CHANGE table rows
- List several ways to READ data and produce reports

Figure 2-30 Unit Summary

Notes:

Unit 3 . What Does An Administrator Do in DB2 UDB?

What This Unit Is About

This unit describes the responsibilities of System and Database Administrator.

What You Should Be Able to Do

After completing this unit, you should be able to:

- List some of the tasks performed by DB2 UDB Administrators
  - Installing
  - Creating
  - Binding
  - Maintaining

How You Will Check Your Progress

Accountability:

- Checkpoint questions

Unit Objectives

After Completing this unit, you should be able to :

- List some of the tasks performed by DB2 Administrators
- What Does a Database Administrator Do in DB2 UDB?
- Indexes
- Views
- Drop
- Locking
- Security

Figure 3-1 Unit Objectives

Notes:

Many types of objects can be created in DB2 UDB: database, table spaces, tables, indexes, views, and synonyms, to name a few.

3.1 What Does A System Administrator Do in DB2 UDB?

What does a System Administrator Do in DB2 UDB?

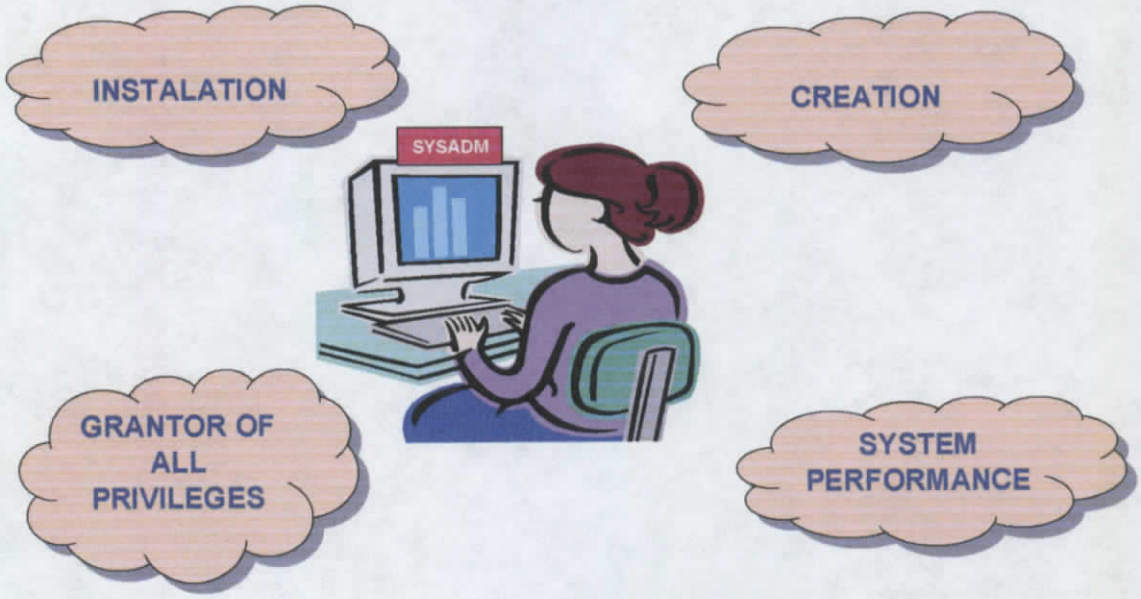


Figure 3-2 What does a System Administrator Do in DB2 UDB?

Notes:

System administrators may administer many DB2 UDBs. They are responsible for creating the system level objects, and the granting of authorities on the system level objects. They are also responsible for the overall performance of DB2 UDB.

IBM DB2 UDB Family

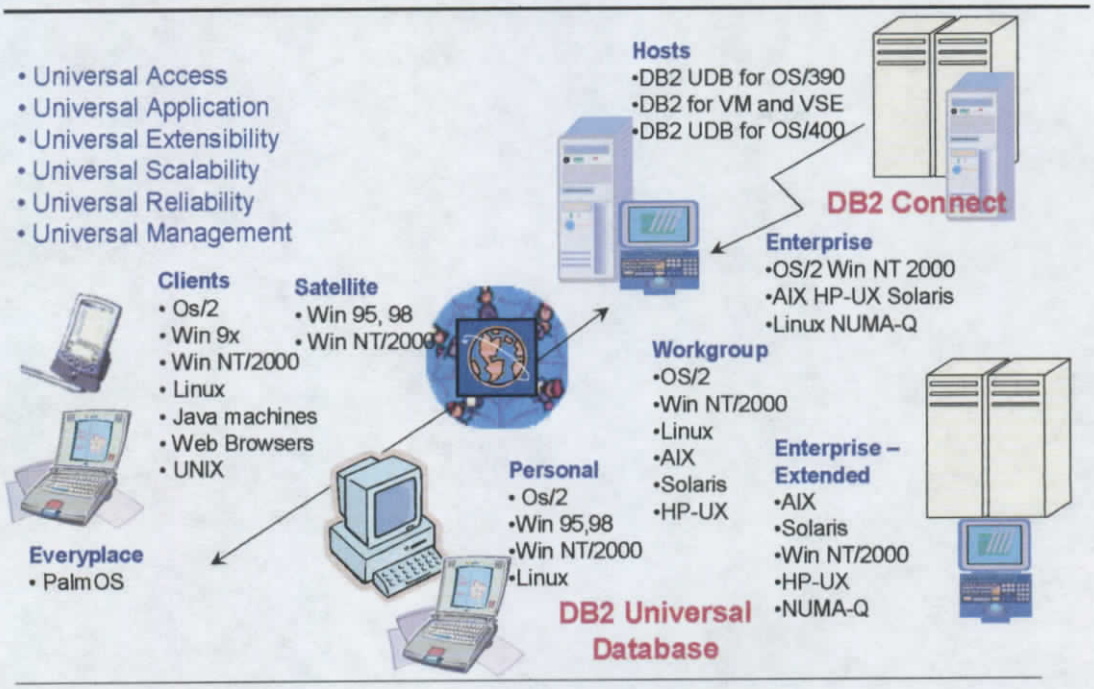


Figure 3-3 IBM DB2 UDB Family

Notes:

DB2 UDB Everyplace

Is a mobile database solution targeted at administration of DB2 UDB Everyplace Clients on hand-held devices.

DB2 UDB Satellite Edition

Is an RDBMS engine with a small footprint that is intended for occasionally connected systems. It will not accept incoming database requests and the maintenance is managed centrally through the Satellite Control Center.
(Windows 32-bit O/S only)

DB2 UDB Personal Edition

Is a full RDBMS engine for an individual machine and can not accept incoming remote calls. It is available on Windows, OS/2 and Linux.

DB2 UDB Workgroup Edition

Is a full RDBMS engine designed for use in a LAN environment. It provides support for both remote and local clients. It is also able to support Internet clients via Net.Data and Java. It is available on OS/2, Windows NT/2000, and UNIX.

DB2 UDB Enterprise Edition

Provides all the function of the Workgroup Edition, plus the ability to connect to host databases using DRDA. It is available on OS/2, Windows NT/2000, and UNIX.

DB2 UDB Enterprise-Extended Edition

Contains all the function of the Enterprise Edition. It also provides the ability for a database to be partitioned across multiple machines of the same platforms. It is designed for applications where the database is simply too large for a single machine. It is available for Windows NT/2000, AIX, HP-UX, NUMA-Q, and Solaris.

DB2 UDB OS/390

Is the large server version (mainframe version) of DB2 UDB.

DB2 UDB Runs in a Massively Parallel Processor Environment

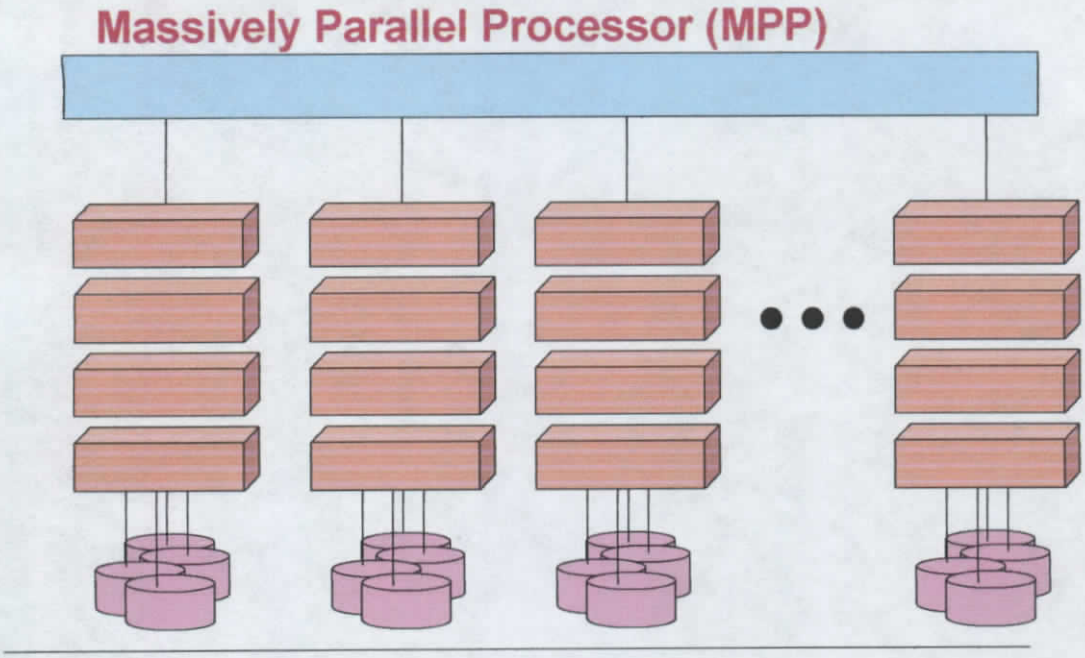


Figure 3-4 DB2 UDB Runs in a Massively Parallel Processor Environment

Notes:

MPPs are needed to deal with the very large databases.

Why Massively Parallel Processors for VLDBs?

Basic Math
Pool holds 1,000,000 gallons
Super fires hose fills at 200 gallons per hour
200*24 (hours per day) = 4800 gallons per day
1,000,000/4800 = 209 days to fill the pool

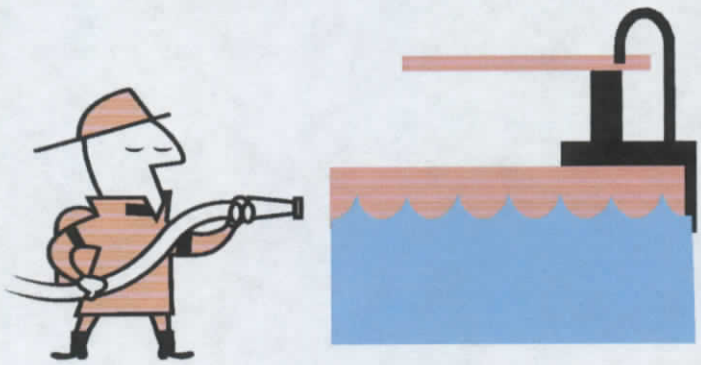


Figure 3-5 Why Massively Parallel Processors for VLDBs?

Notes:

Very Large Databases (VLDB)

- Today there are 200 terabyte databases in production.
- The pool represents a 1 pedabyte (1,000,000 gigabytes) tables. Each gallon represents a gigabyte of data. Assume that a single processor can process 200 gigabytes an hour.

How can we fill the pool faster?

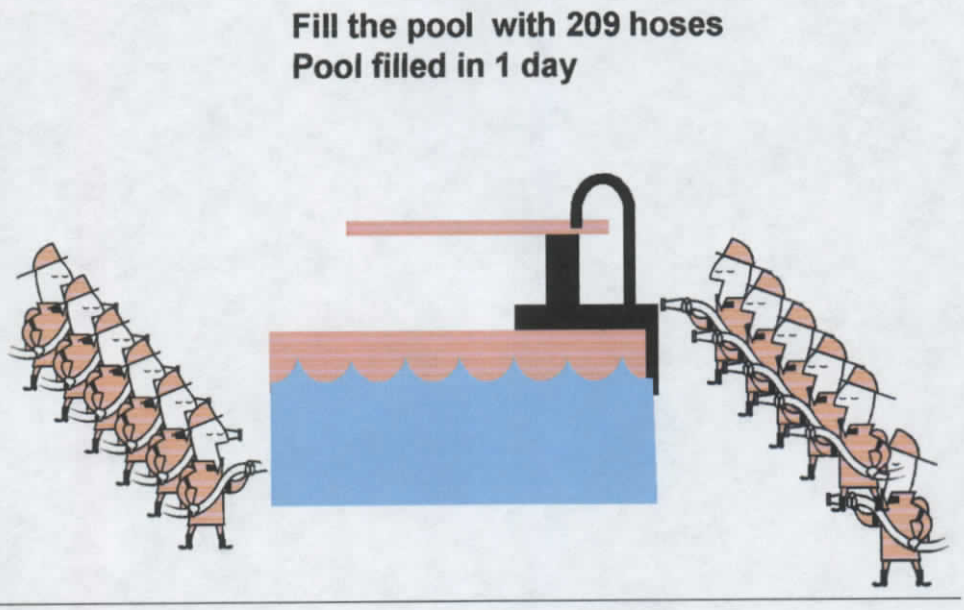


Figure 3-6 How can we fill the pool faster?

Notes:

Two hundred and nine input streams would allow us to fill the pool in about a day (think of this as parallel I/O).

What Happens when We Have to clean the pool?

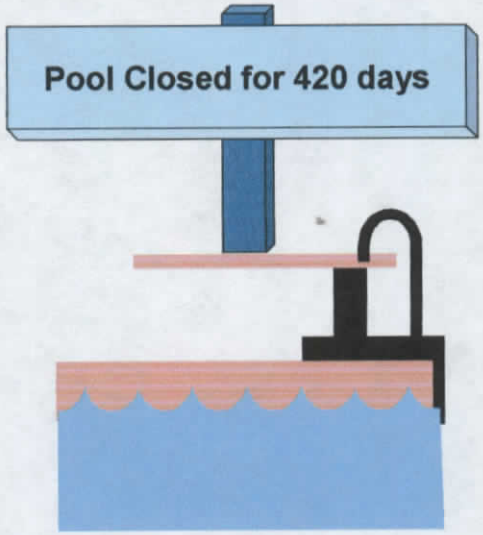


Figure 3-7 What Happens when We Have to clean the pool?

Notes:

To 'clean' the pool takes 209 days to empty the pool, time to clean it and 209 days to refill it. Every so often data must be removed, rearranged and reloaded in a database. This is called a REORG.

Installation – Physical Install

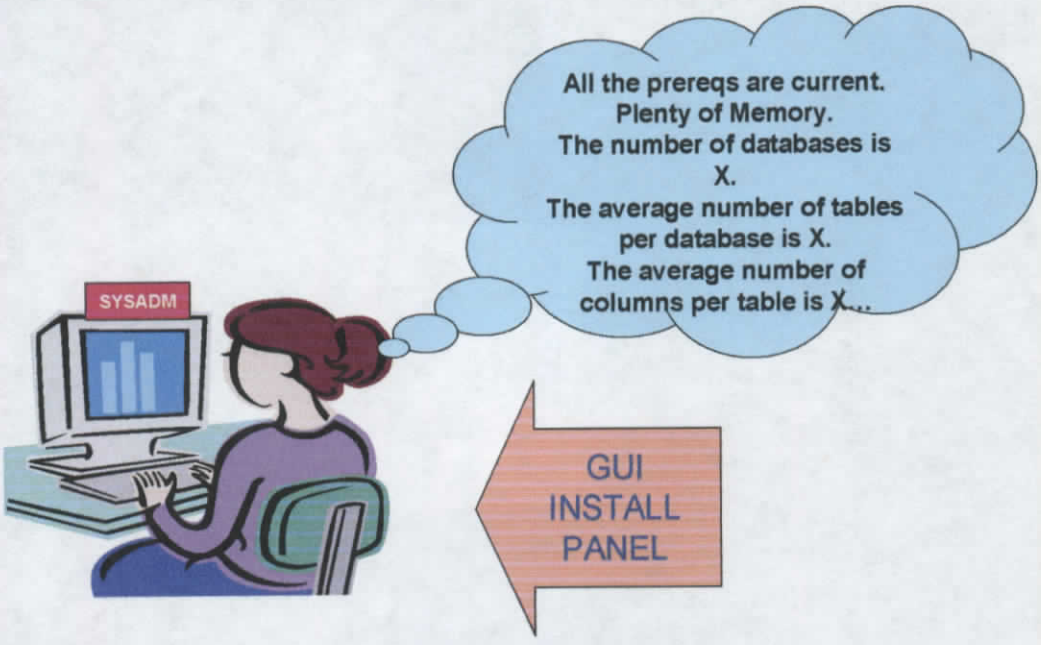


Figure 3-9 Installation – Physical Install

Notes:

- The SYSADM, (or system programmer) must ensure that all supporting software is at the proper release level for the current release of DB2 UDB.
- Calculations must be done to determine the amount of storage needed to store and manipulate the projected data volumes.
- SYSADM may need to gather information that will affect their installation, such as how many databases we'll be creating, the average number of expected tables per database, the maximum number of users, and so forth.

Create system Level Object

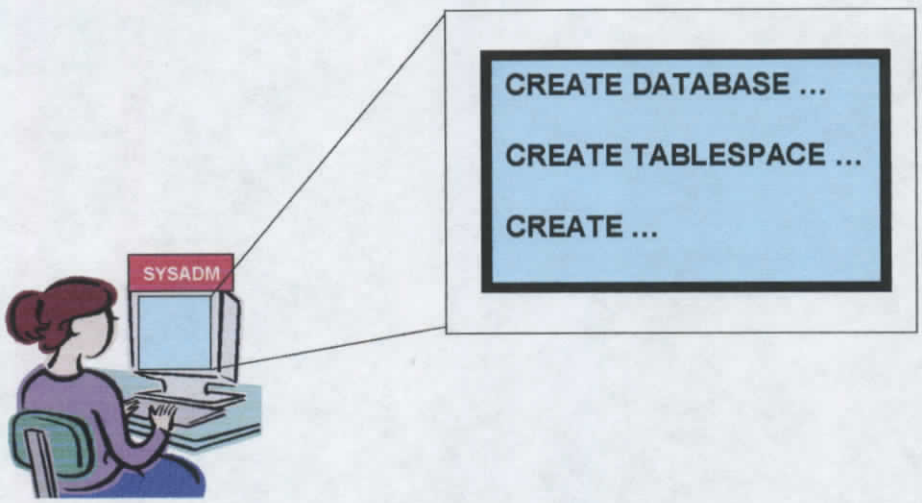


Figure 3-10 Create system Level Object

Notes:

All DB2 UDB objects are created via SQL, or, through panels that generate the SQL under the covers.

Using the DB2 UDB Control Center

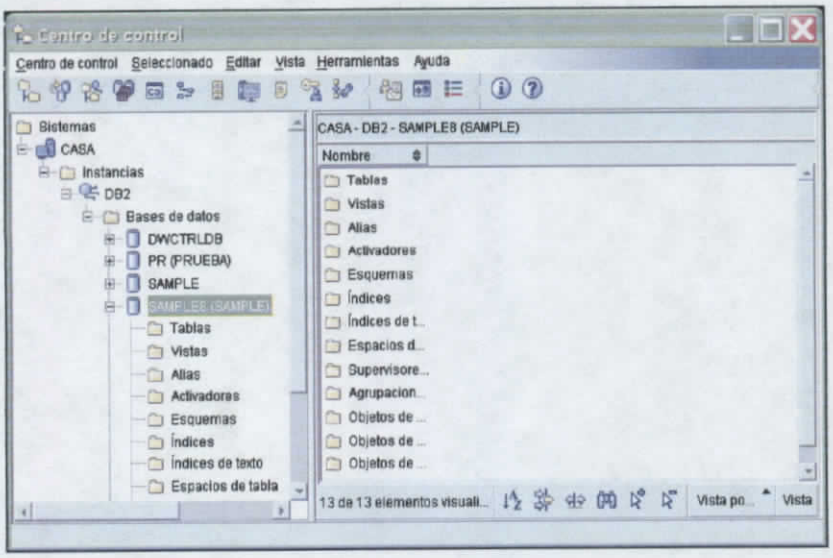


Figure 3-11 Using the DB2 UDB Control Center

Notes:

To create a new object the administrator right clicks an object of the same type and clicks CREATE from the drop down menu. Objects can be created with straight SQL, or created by a wizard and mouse clicks.

Grant Privileges

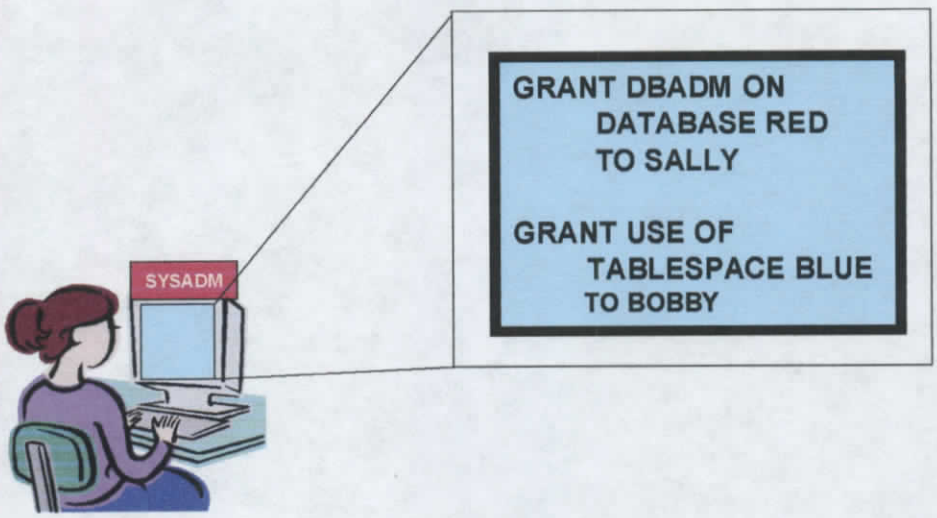


Figure 3-12 Grant Privileges

Notes:

Once the databases have been created the SYSADM can grant needed privileges to other users.

In the UNIX/Intel environment a user must have the USE privilege to create tables.

3.2 What Does A Database Administrator Do in DB2 UDB?

What Does a Database Administrator Do in DB2 UDB?

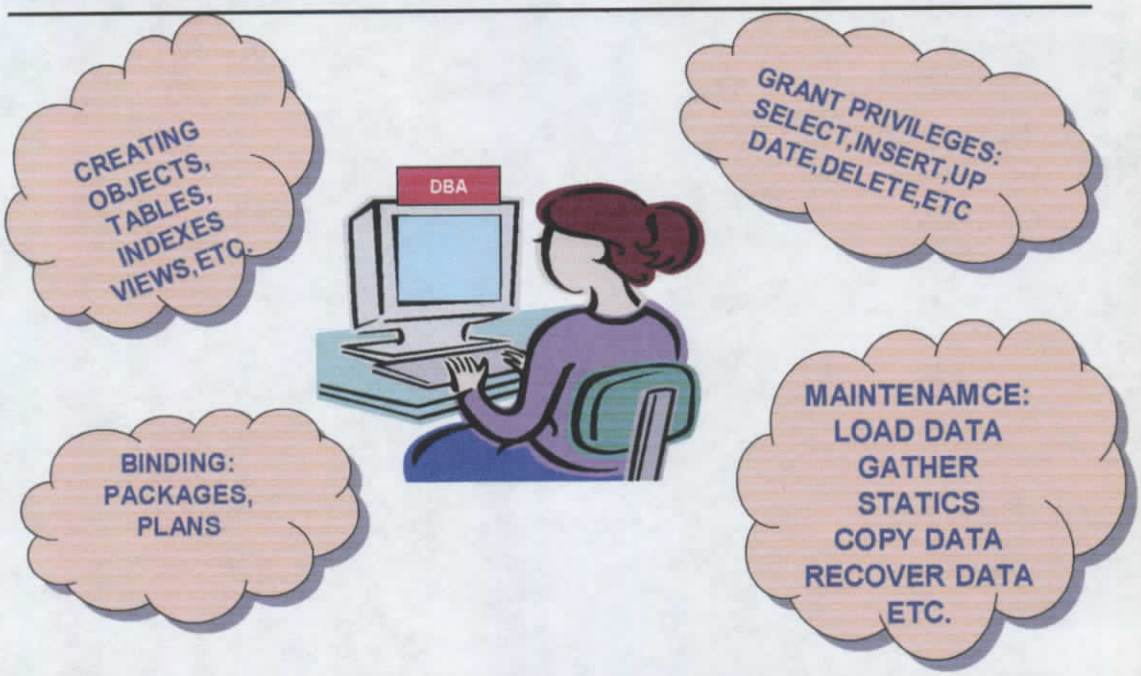


Figure 3-13 What Does a Database Administrator Do in DB2 UDB?

Notes:

A DBA creates objects and grants privileges within the scope of the database.

Creating Tables Using the GUI

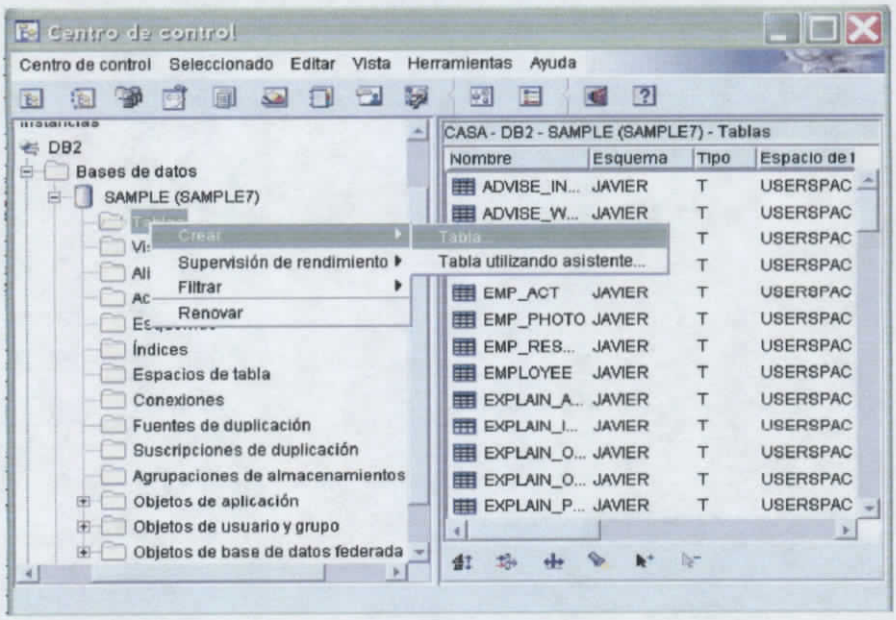


Figure 3-14 Creating Tables Using the Table Wizard

Notes:

To create a table using the Table Wizard we right click a table that is in the database in which we want out table to reside. From the popup menu, click Create, Table Using Wizard.

Name the table

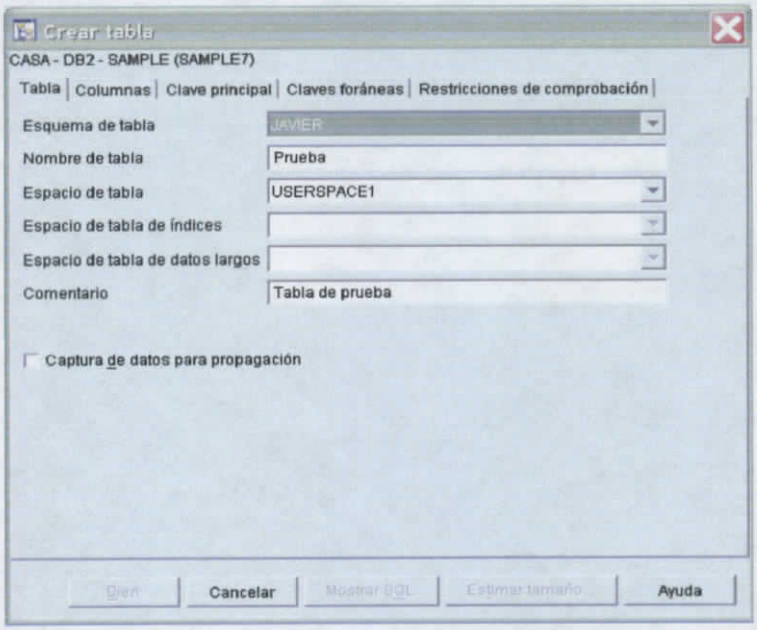


Figure 3-15 Name the table

Notes:

The schema is the owner or creator of the table. Fill in the blanks and click Next.

Choose Columns

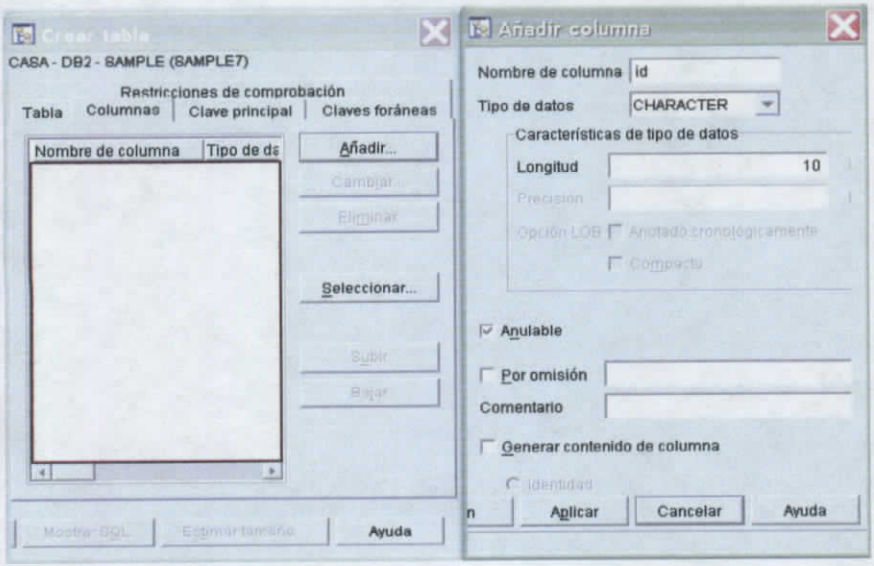


Figure 3-16 Choose Columns

Notes:

Predefined column categories help speed the table building process. Each category has sets of the kinds of column commonly used for data of that category.

Click a category, click the column name and click the right arrow to move it to the Columns to create pane. Columns from any number of categories can be chosen.

Don't like a column's name, click Columns to create pane and change it in the Column name pane.

Defining a Primary Key

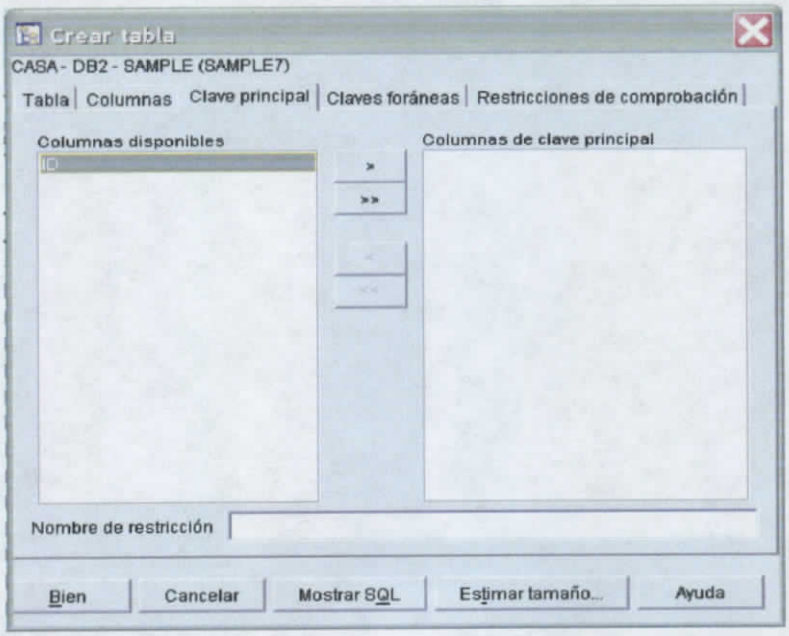


Figure 3-17 Defining a Primary Key

Notes:

The next step in creating our table is to define the primary key. Click the column we wish to be the primary key and click the right arrow button.

If the primary key is to be comprised of more than one column, we repeat the process; otherwise, we click Next.

Defining a Foreign Key

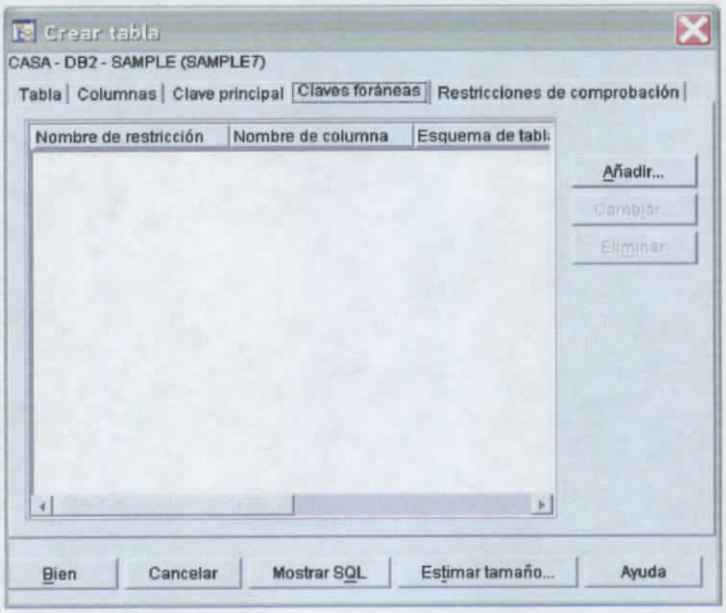


Figure 3-18 Defining a Foreign Key

Notes:

Review SQL - Finish

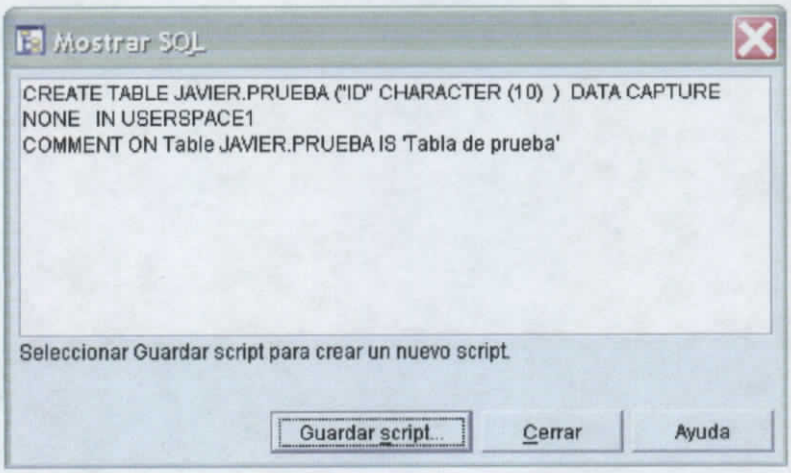


Figure 3-19 Review SQL - Finish

Notes:

Click Show SQL from the summary option and we can review the SQL syntax the Wizard wrote on our behalf.

Alter Table

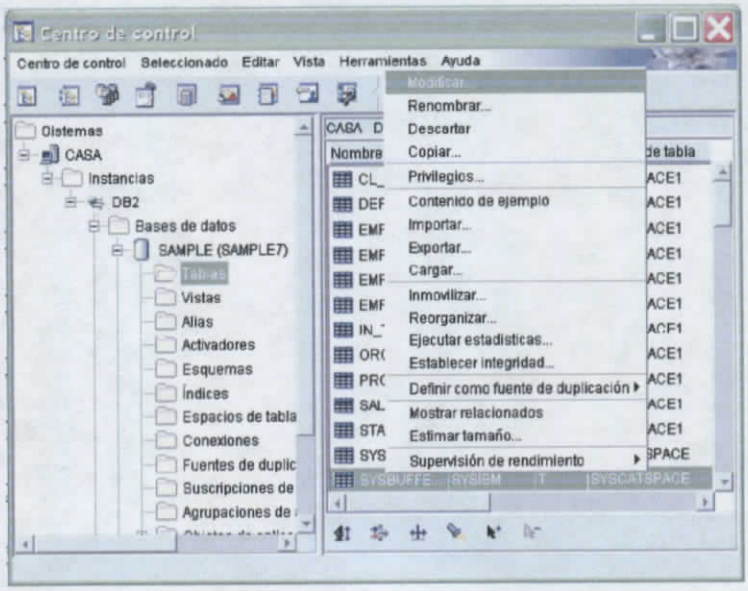


Figure 3-20 Alter Table

Notes:

Changed to a table's definition can be made by right-clicking the table's name and clicking Alter from the menu.

Add Business Rules, Foreign Key, and so on

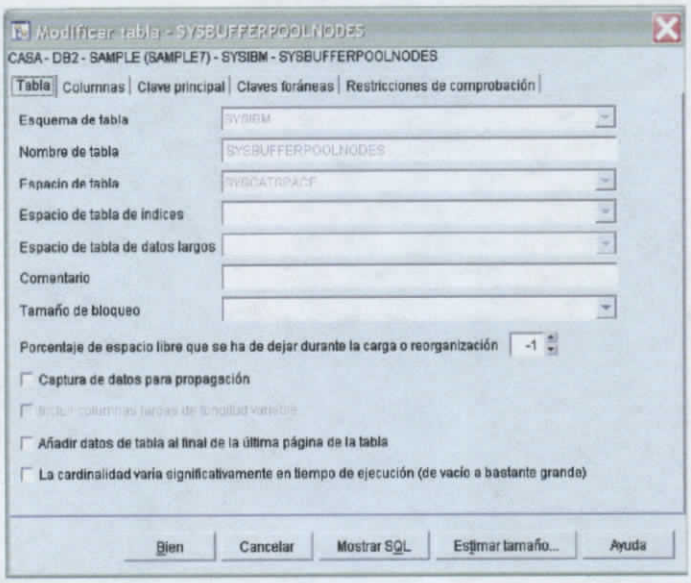


Figure 3-21 Add Business Rules, Foreign Key, and so on

Notes:

Enter the table's name and choose options from the various tabs.

3.3 Indexes

Tables without Indexes – From the table COMPANY.DIRECTORY

Last Name	First Name	Address	Ph
SMITH	DIANA	2253 CEVALLOS	
GAINES	LOIS	6094 SUCRE	
BROWN	TERRY	45 RIVER ST	
ADAMS	SANDY	14 GEORG	
FRIDRICH	JOHN	240 BOS	413-435
MITCHELL	TED	9	420-230
JONES	LARRY	1	412-857
WISE	SARA	6	555-652
JONES	LAVERNE	45 F	555-6987
MITCHELL	JEAN	6094	555-987
ADAMS	BOBBY	14 GEORG	555-647
JONES	LARRY	240 BOSTON	555-142
SMITH	GRACE	983LA	555-874
CALDWELL	SIMONE	122 42ND ST	555-536

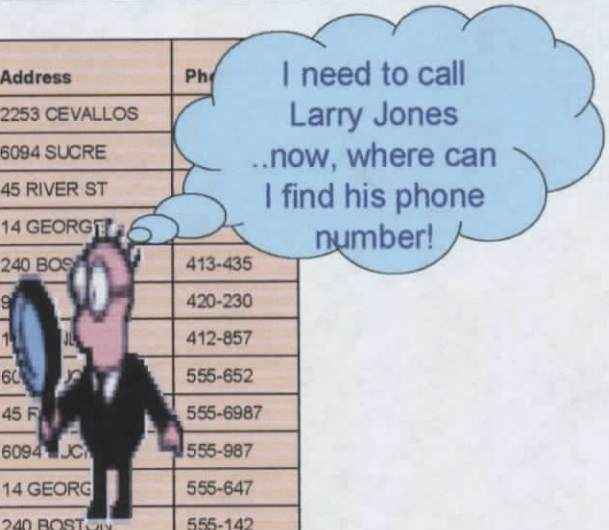


Figure 3-22 Tables without Indexes – From the table COMPANY.DIRECTORY

Notes:

With no indexes every row must be read. Remember, row order is NOT guaranteed as rows are inserted, updated, and deleted from tables.

An Index Is:

A DB2 tool used to
Located a row or rows
That meet a particular
condition

An ordered list of values with
One or more pointers
(called RIDs) to where
These values are found
Within the table.

Figure 3-23 An Index Is:

Notes:

DB2 UDB use indexes for three reasons:

- To speed up access to data
- To ensure uniqueness
- Clustering

Creating Index

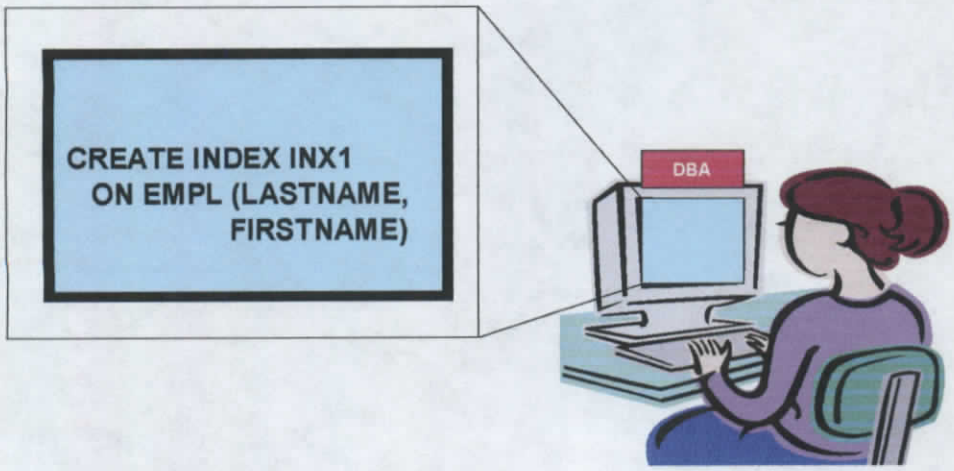


Figure 3-24 Creating Index

Notes:

There are many additional index parameters that may be specified when an index is created. These parameters are discussed in the Database Administration Workshop.

Creating an Index

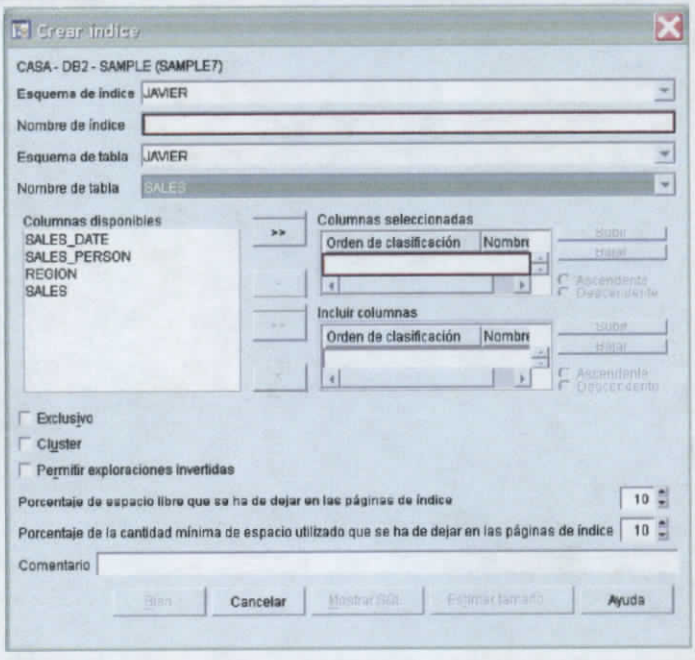


Figure 3-25 Creating an Index

Notes:

Name the index, select the columns, their sort sequence, whether or not the index is clustered or unique, etc.

3.4 Views

Views

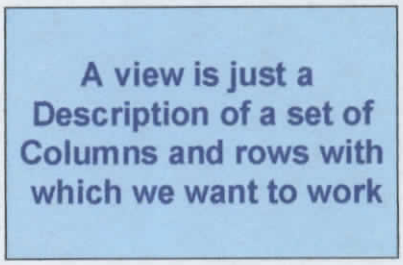


Figure 3-26 Views

Notes:

A view is a way to "look" at data that exists in one or more tables.

Creating a View

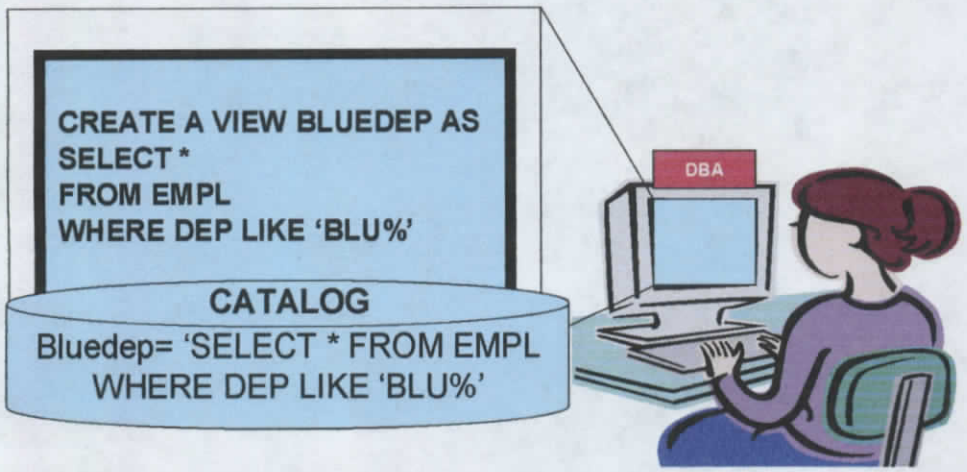


Figure 3-27 Creating a View

Notes:

When a view is created, its definition is stored in the DB2 UDB catalog.

When the view name is used in an SQL statement, it is like telling DB2 UDB to run my query against the set of data defined by the view's select statement.

Using a View

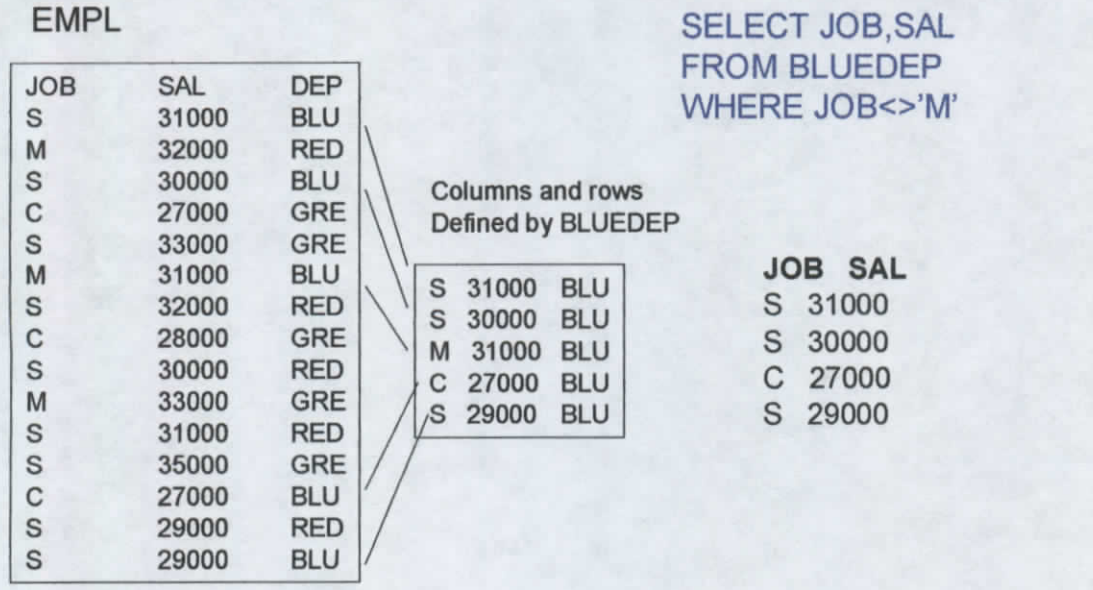


Figure 3-28 Using a View

Notes:

In this example, using the view BLUEDEP tells DB2 UDB to apply the SELECT statement to the set of rows whose DEP value begins with the letters BLU.

Updating a Through a View

EMPL

JOB	SAL	DEP
S	31000	BLU
M	32000	RED
S	30000	BLU
C	27000	GRE
S	33000	GRE
M	31000	BLU
S	32000	RED
C	28000	GRE
S	30000	RED
M	33000	GRE
S	31000	RED
S	35000	GRE
C	27000	BLU
S	29000	RED
S	29000	BLU

```
UPDATE BLUEDEP
SET SALARY =
SALARY + 1000
WHERE JOB <> 'M'
```

EMPL

JOB	SAL	DEP
S	32000	BLU
M	32000	RED
S	31000	BLU
C	27000	GRE
S	33000	GRE
M	31000	BLU
S	32000	RED
C	28000	GRE
S	30000	RED
M	33000	GRE
S	31000	RED
S	35000	GRE
C	28000	BLU
S	29000	RED
S	30000	BLU

Figure 3-29 Updating a Through a View

Notes:

In this example, DB2 UDB will update only those rows in the EMPL table whose DEP value begins with the letters BLU if and only if their JOB value is not the letter M.

3.5 Drop

The DBA Cleans Up

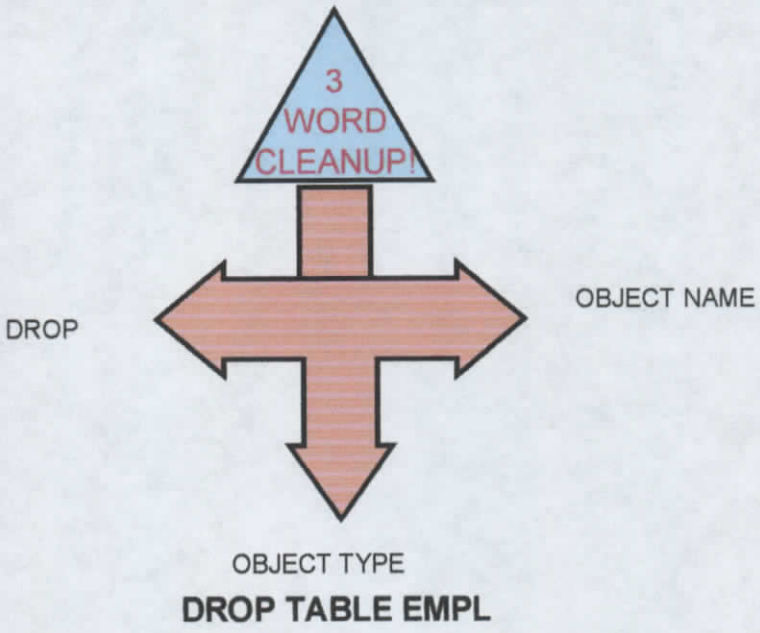


Figure 3-30 The DBA Cleans Up

Notes:

It is very easy to remove a DB2 UDB object (if you have the authority to do so).

Clean up Using the GUI

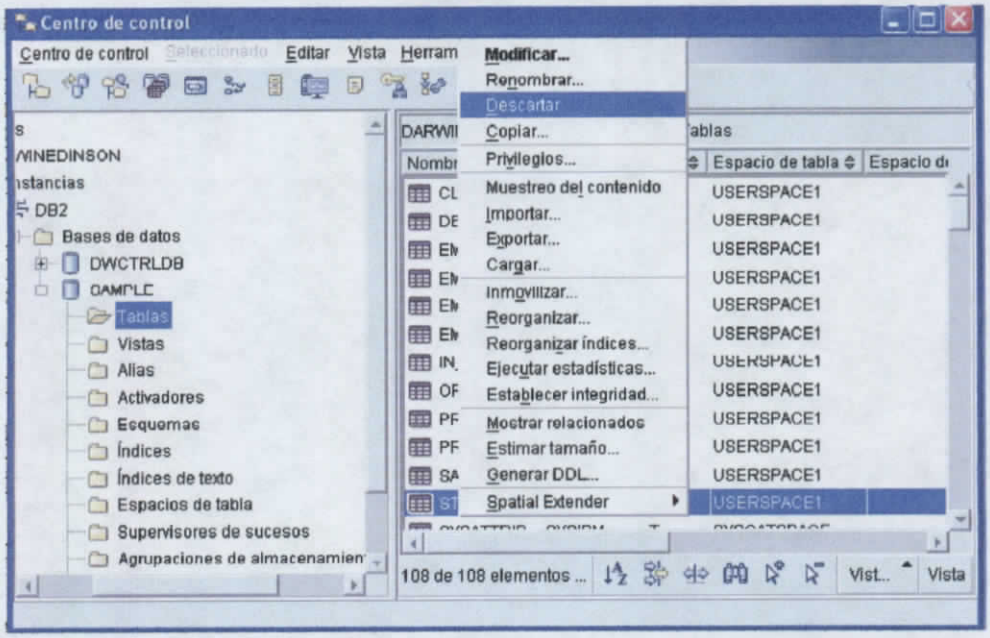


Figure 3-31 Clean up Using the GUI

Notes:

Right-click the object to be removed and click Drop from the menu.

DROP Dependencies

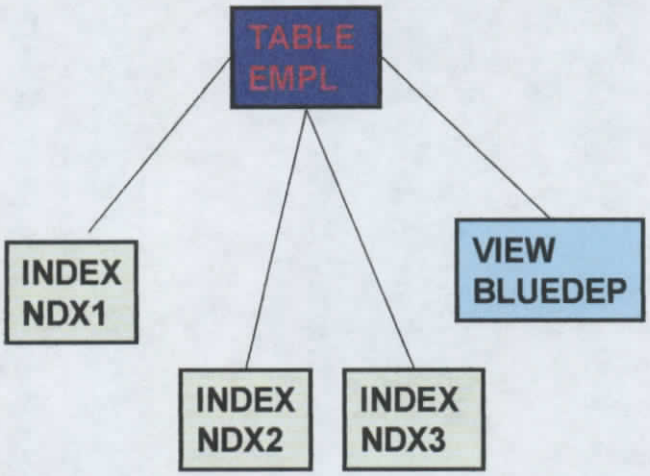


Figure 3-32 DROP Dependencies

Notes:

Dropping the table causes DB2 UDB to 'cascade' the drop to all objects dependent on the table (indexes, views, synonyms).

3.6 Locking

Concurrency – Table Level (1 of 6)

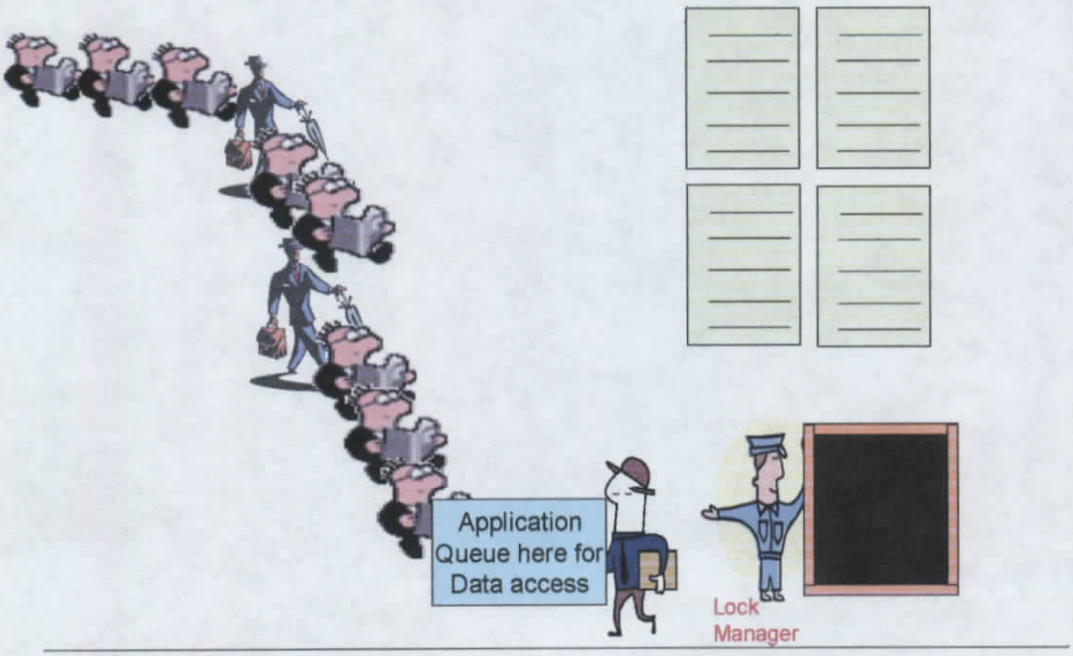


Figure 3-33 Concurrency – Table Level (1 of 6)

Notes:

If the DBA tells DB2 UDB to maintain locking at the table level, conceptually, DB2 UDB's lock enforcer will look at the first application in the queue to see what it wants to do to the data in the table.

In the above example, the first application wants to READ the data. The lock enforcer sets up a sign indicating that the gate to the table is open for reading purposes.

Concurrency – Table Level (2 of 6)

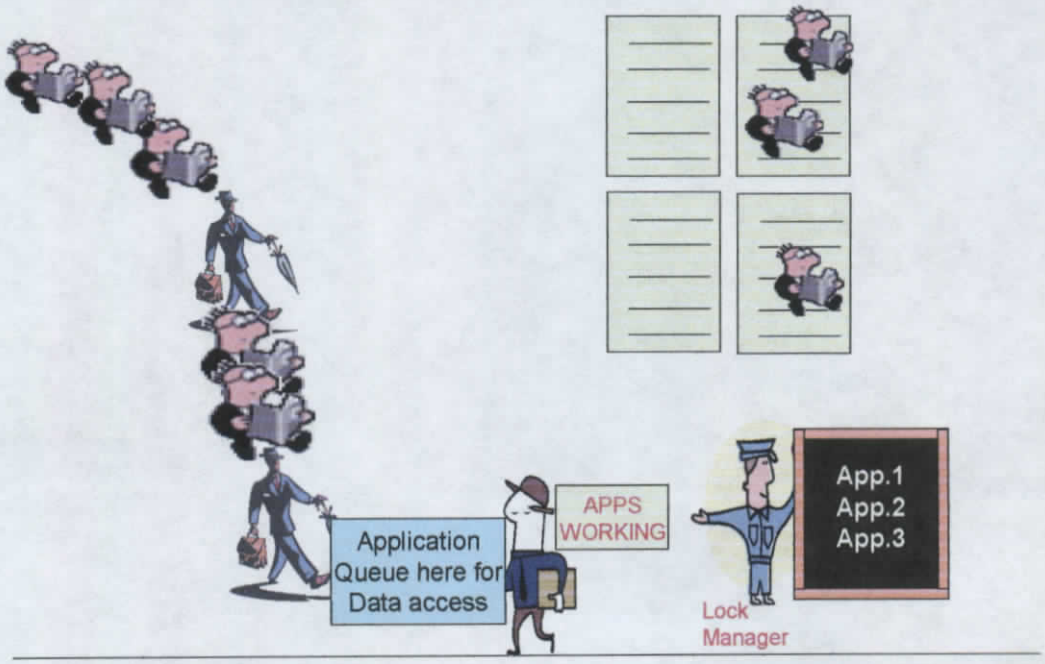


Figure 3-34 Concurrency – Table Level (2 of 6)

Notes:

Each consecutive application in the queue that wants to read DB2 UDB will let it into the table. DB2 UDB keeps track of which applications it has let into the table.

In the above example, the first three applications in the queue wanted to read only so DB2 UDB let them into the table. The fourth application wanted to make changes to the table so it and the applications behind it in the queue must wait until all the readers leave the table.

Concurrency – Table Level (3 of 6)

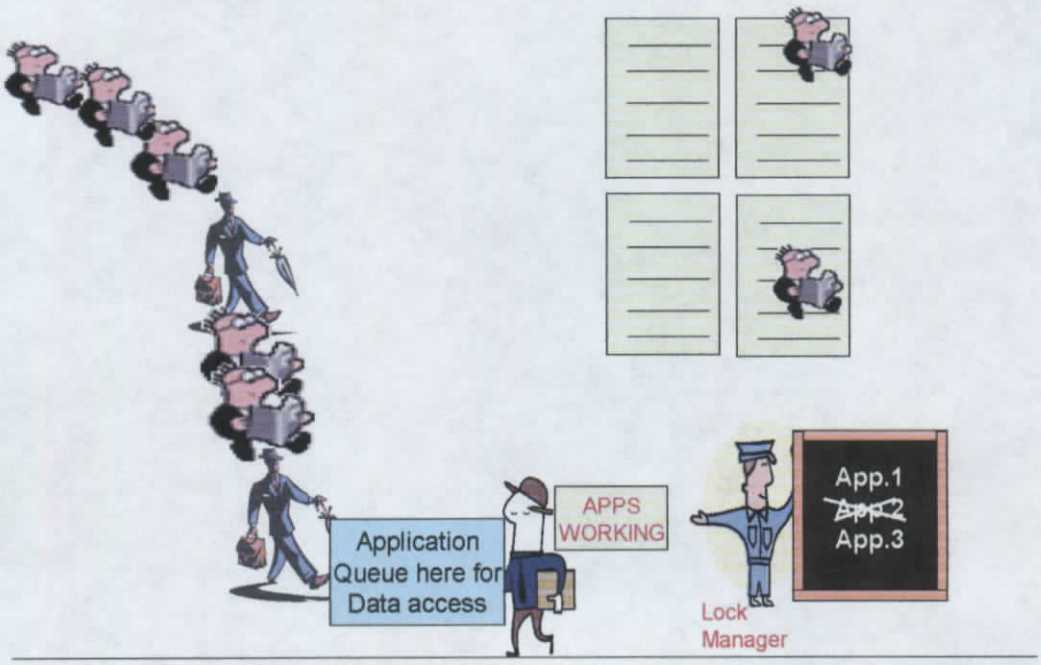


Figure 3-35 Concurrency – Table Level (3 of 6)

Notes:

As each of the reading applications finishes their work, the lock manager crosses them off the black board.

Concurrency – Table Level (4 of 6)

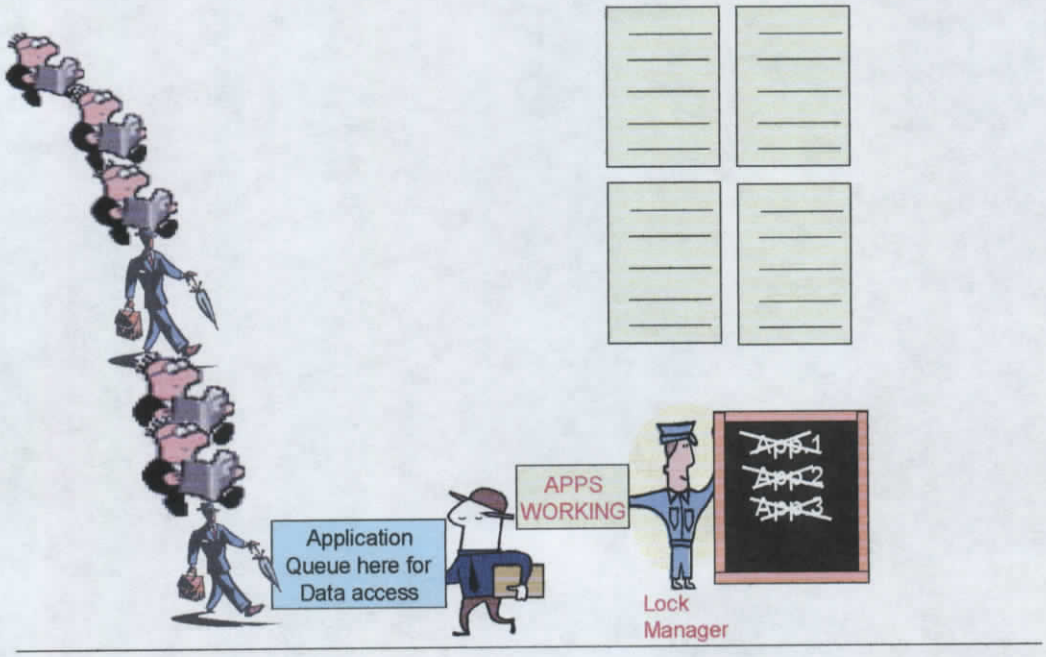


Figure 3-36 Concurrency – Table Level (4 of 6)

Notes:

When the last read application leaves, the lock enforcer will change the access sign to the table. Applications that change data require EXCLUSIVE access to that data.
Since the DBA told DB2 UDB that locking was to be maintained at the table level for this table, only one lock enforcer was 'hired'. The lock enforcer will let in only one change application at a time.

Concurrency – Table Level (5 of 6)

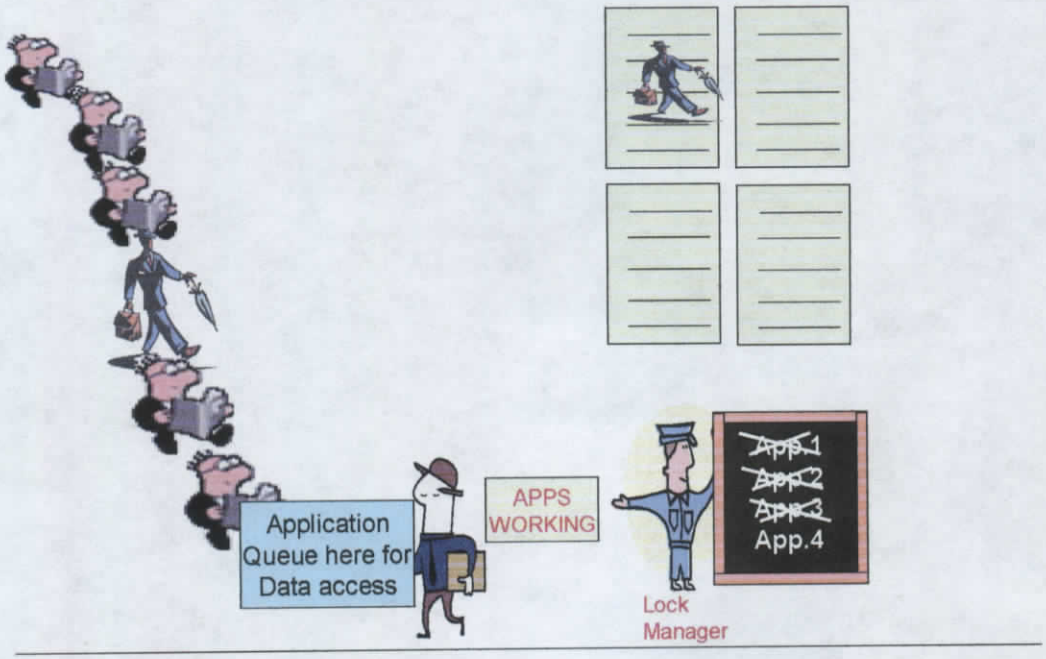


Figure 3-37 Concurrency – Table Level (5 of 6)

Notes:

Application 4 is making its changes while all other applications remain in the queue.

Concurrency – Table Level (6 of 6)

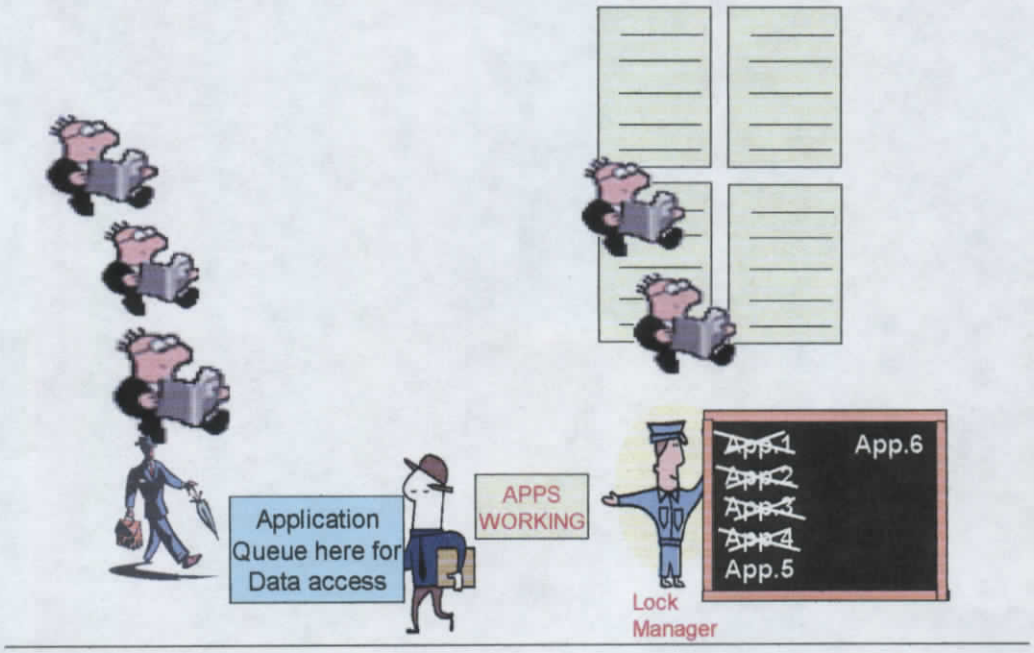


Figure 3-38 Concurrency – Table Level (6 of 6)

Notes:

Once the change application has completed its work, the lock enforcer checks what the next application in the queue wants to do. Since it wants to read, as does the next application in the queue, both applications are let into the table. Application 7 wants to make changes to the data (requiring EXCLUSIVE access) so it and subsequent applications remain in the queue.

Concurrency – Row Level

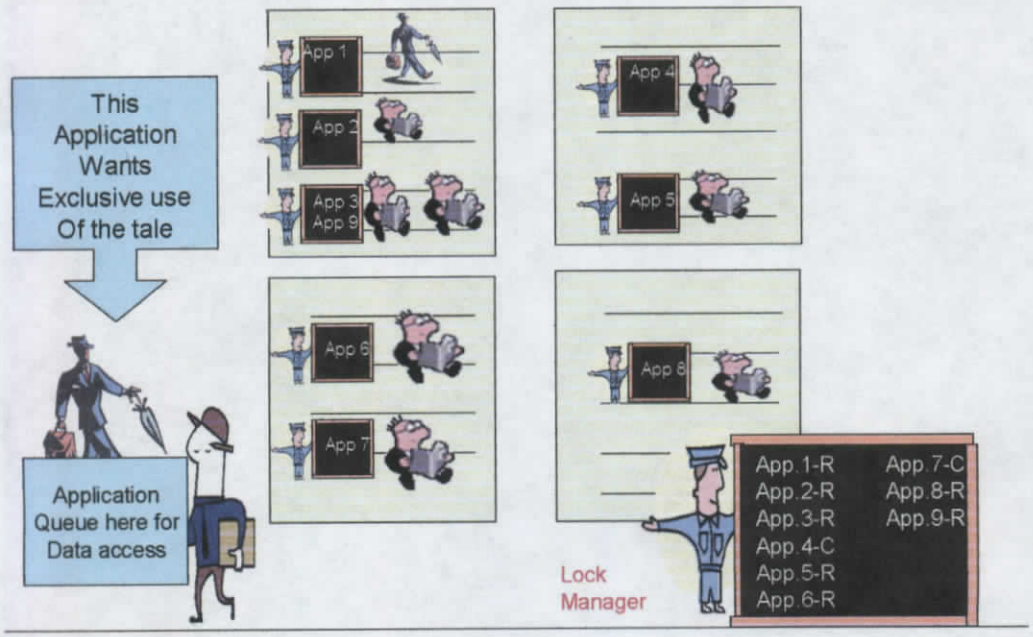


Figure 3-39 Concurrency – Row Level

Notes:

This example assumes that the DBA choose ROW level locking for this table. When ROW level locking is in place, it is like hiring a lock enforcer for the table and one lock enforcer for each row.

The example shows that there is one application waiting outside the table because it demands EXCLUSIVE access to the table; however, within the table, all applications are working because, for this example, there is no conflict at the row level. Conflict at the row level would occur if a change application and any other application wanted access to the same row at the same time. When this occurs, the lock enforcer gives access to the first application queued for the row.

3.7 Security

To Get to DB2 UDB

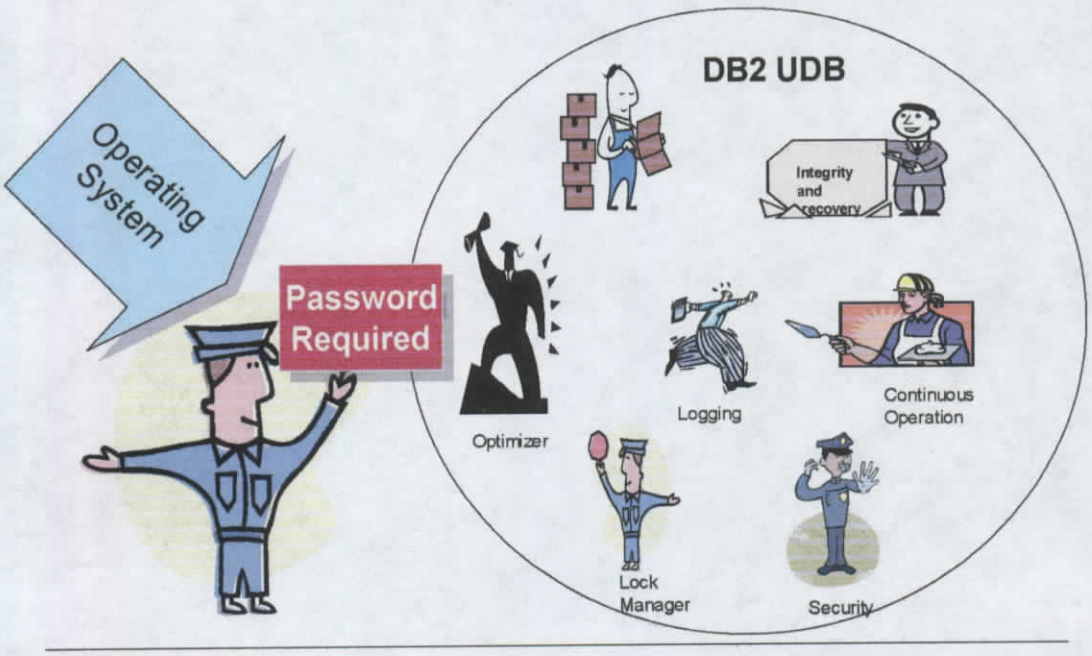


Figure 3-40 To Get to DB2 UDB

Notes:

Must get through the operating systems security manager before we can get to DB2 UDB (usually requires an ID and password).

DB2 Security

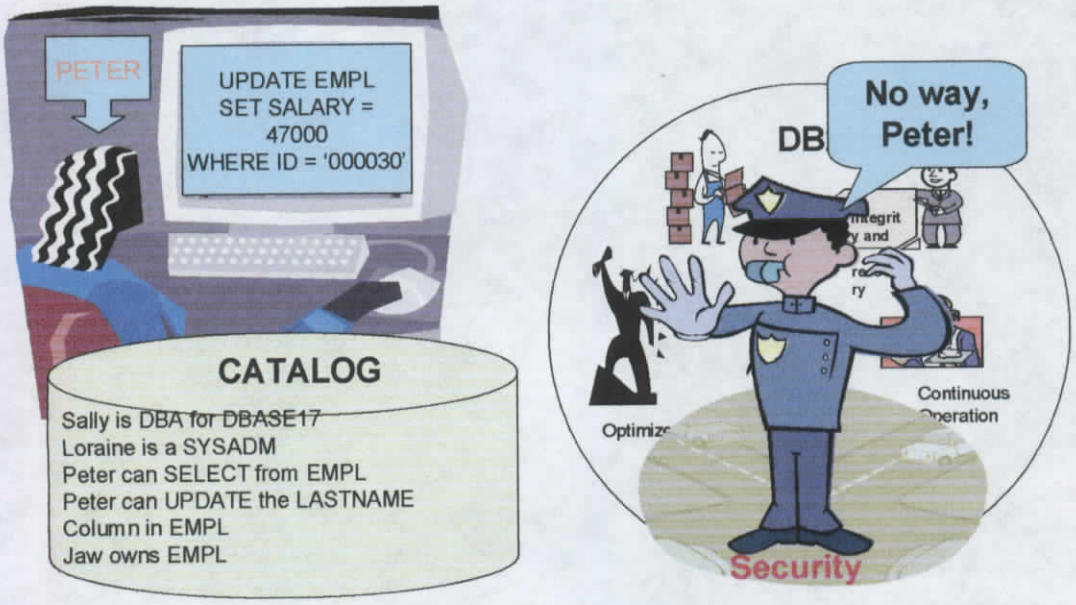


Figure 3-41 DB2 Security

Notes:

Before a user can access a DB2 UDB object, DB2 UDB goes to the catalog and checks to see if the user has the authority to do what they are trying to do.

Privileges

- IMPLICIT – Owner's privileges
- EXPLICIT – Specifically GRANTed privileges
- INDIRECT – Group

Figure 3-42 Privileges

Notes:

These are the three ways to acquire DB2 UDB privileges.

Owner's (Implicit) Table Privileges



Figure 3-43 Owner's (Implicit) Table Privileges

Notes:

DML – Data Manipulation Language, the set of SQL statements which contain SELECT, INSERT, UPDATE, and DELETE.

Explicit Table Privileges

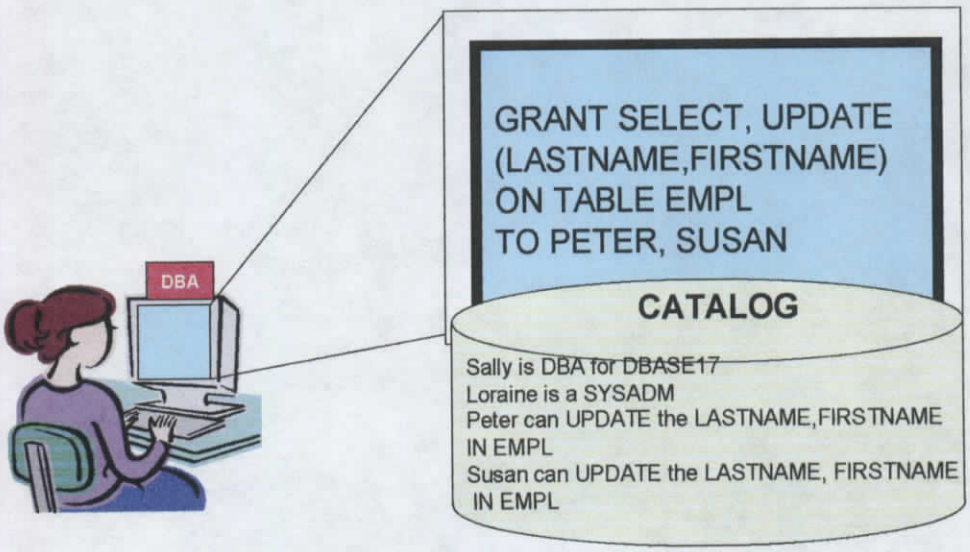


Figure 3-44 Explicit Table Privileges

Notes:

Explicit table privileges are given via a GRANT SQL statement.

Indirect Privileges-Group

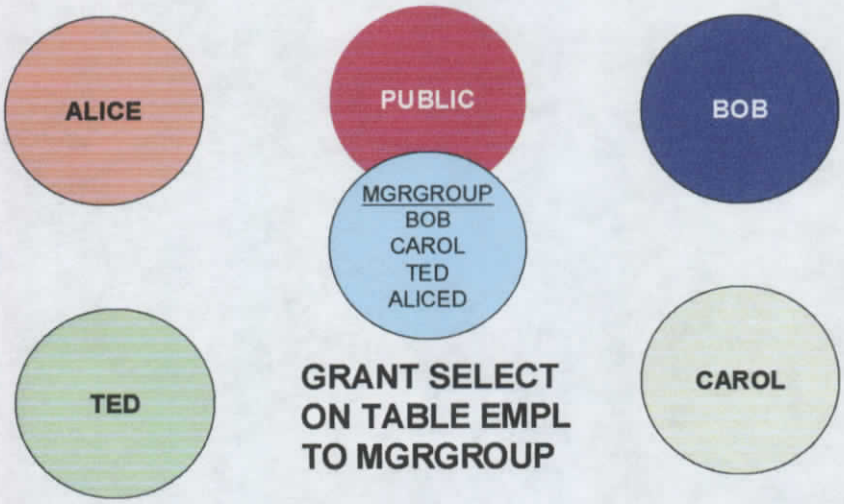


Figure 3-45 Indirect Privileges-Group

Notes:

Everyone with access to DB2 UDB is part of the PUBLIC group. To GRANT a privilege to everyone, GRANT it to PUBLIC.

A privilege can be GRANTED to a group level ID (MGRGROUP) which will make that privilege available to each member of the group, without having to grant the privilege to specific user IDs.

Administrative Authorities

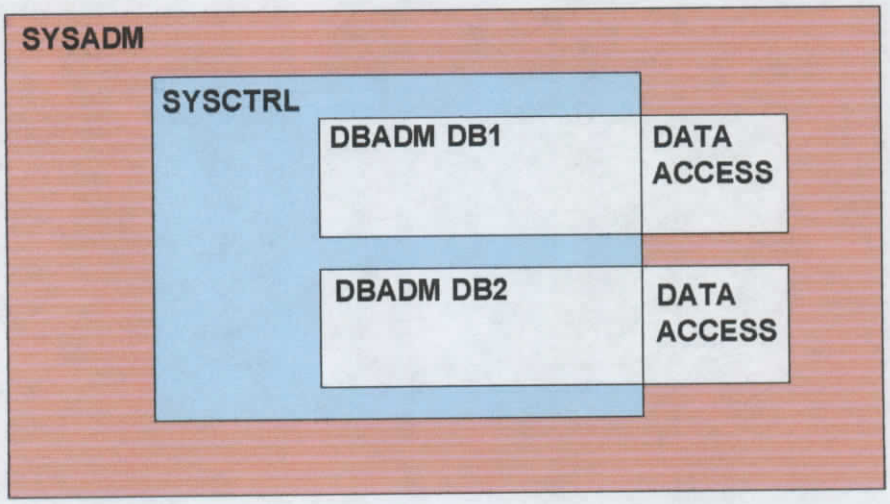


Figure 3-46 Administrative Authorities

Notes:

An administrative authority conveys a set of privileges.

Views and Security

EMPL

JOB	SAL	DEP
S	31000	BLU
M	32000	RED
S	30000	BLU
C	27000	GRE
S	33000	GRE
M	31000	BLU
S	32000	RED
C	28000	GRE
S	30000	RED
M	33000	GRE
S	31000	RED
S	35000	GRE
C	27000	BLU
S	29000	RED
S	29000	BLU

```
CREATE VIEW BLUEDEP AS
SELECT *
FROM EMPL
WHERE DEP = 'BLU'
```

```
GRANT SELECT,
UPDATE (SALARY)
ON BLUEDEP
TO CLERK
```

Figure 3-47 Views and Security

Notes:

Views allow a more granular control of privileges in DB2 UDB.

Revoke

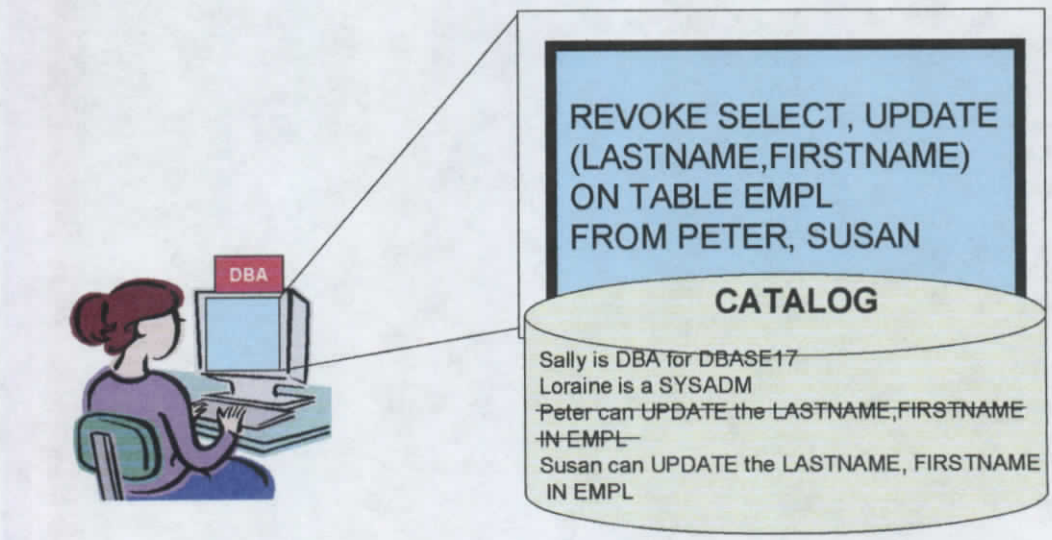


Figure 3-48 Revoke

Notes:

REVOKE removes privilege(s) from the catalog.

Unit Summary

Since completing this unit, you should be able to:

- List some of the tasks performed by DB2 Administrators
- What Does a Database Administrator Do in DB2 UDB?
- Indexes
- Views
- Drop
- Locking
- Security

Figure 3-49 Unit Summary

Notes:

Checkpoint

Exercise — Unit 3 Checkpoint

1. Who is responsible for creating DB2 UDB databases?

2. What is a DB2 UDB index?

3. List a couple of reasons indexes are used in DB2 UDB?

4. What is a view?

5. What must be true if USER1 can successfully update the ADDRESS column in table USER7.TAB_1?

6. List some of the things a table's owner can do to/with their table.

7. Create a view TEMP with the table TEST_EMP

   EMPNO
   LASTNAME
   SALARY

8. Insert the following data into the TEMP view

   Employee number: 000500
   Last name: 'SANCHEZ'
   Salary: $ 1000

9. Clean you TEST_EMP table

Unit 4 Simple SQL Queries

What This Unit Is About

The SELECT statement is used to retrieve data from a table. In this unit, we will discuss the format and syntax of basic SELECT statements. In subsequent units, we will be looking at more complex SELECT statements.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Specify four clauses (SELECT, FROM, WHERE, ORDER BY) used in SELECT statements, and the purpose of each.
- Retrieve data from a table based on column values.
- Sequence the result into the desired order.

How You Will Check Your Progress

- Checkpoint questions
- Machine labs

Objectives

After completing thus unit, you should be able to :

- Syntax of SQL SELECT statements
- Selection of entire Tables
- Ordering The result table
- Selection of specific columns
- Selection of specific rows
  - Comparison with alphanumeric columns
  - Comparison with numeric columns
  - Multiple conditions
- Use the Keywords
  - DISTINCT
  - LIKE
  - IN
  - BETWEEN

Figure 4-1 Summary

Notes:

4.1 Simple SQL Queries

Employee Table (Part 1)

EMPNO	FIRSTNME	MIDINT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
10	CHRISTINE	I	HAAS	A00	3978	01/01/1965
20	MICHAEL	L	THOMPSON	B01	3476	10/10/1973
30	SALLY	A	KWAN	C01	4738	05/04/1975
50	JOHN	B	GEYER	E01	6789	17/08/1949
60	IRVING	F	STERN	D11	6423	14/09/1973
70	EVA	D	PULASKI	D21	7831	30/09/1980
90	EILEEN	W	HENDERSON	E11	5498	15/08/1970
100	THEODORE	Q	SPENSER	E21	972	19/06/1980
110	VINCENZO	G	LUCCHESSI	A00	3490	16/05/1958
120	SEAN		O'CONNELL	A00	2167	05/12/1963
130	DOLORES	M	QUINTANA	C01	4578	28/07/1971
140	HEATHER	A	NICHOLLS	C01	1793	15/12/1976
150	BRUCE		ADAMSON	D11	4510	12/02/1972
160	ELIZABETH	R	PIANKA	D11	3782	11/10/1977
170	MASATOSHI	J	YOSHIMURA	D11	2890	15/09/1978

Figure 4-2 Employee Table (Part 1)

Notes:

EMPNO	Employee number
FIRSTNME	First name
MIDINT	Middle initial
LASTNAME	Last name
WORKDEPT	Department number
PHONENO	Phone number
HIREDATE	Hire date

Employee Table (Part 1, Cont'd)

EMPNO	JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
10	PRES	18	F	24/08/1933	52750	1000	4220
20	MANAGER	18	M	02/02/1948	41250	800	3300
30	MANAGER	20	F	11/05/1941	38250	800	3060
50	MANAGER	16	M	15/09/1925	40175	800	3214
60	MANAGER	16	M	07/07/1945	32250	500	2580
70	MANAGER	16	F	26/05/1953	36170	700	2893
90	MANAGER	16	F	15/05/1941	29750	600	2380
100	MANAGER	14	M	18/12/1956	26150	500	2092
110	SALESREP	19	M	05/11/1929	46500	900	3720
120	CLERK	14	M	18/10/1942	29250	600	2340
130	ANALYST	16	F	15/09/1925	23800	500	1904
140	ANALYST	18	F	19/01/1946	28420	600	2274
150	DESIGNER	16	M	17/05/1947	25280	500	2022
160	DESIGNER	17	F	12/04/1955	22250	400	1780
170	DESIGNER	16	M	05/01/1951	24680	500	1974

Figure 4-3 Employee Table (Part 1, Cont'd)

Notes:

EMPNO	Employee number (repeated for readability purpose)
JOB	Job
EDLEVEL	Education level (Ed-level)
SEX	Sex
BIRTHDATE	Date of birth
SALARY	Salary (yearly)
BONUS	Bonus (yearly)
COMM	Commission (yearly)

Employee Table (Part 2)

EMPNO	FIRSTNME	MIDINT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
180	MARILYN	S	SCOUTTEN	D11	1682	07/07/1973
190	JAMES	H	WALKER	D11	2986	26/07/1974
200	DAVID		BROWN	D11	4501	03/03/1966
210	WILLIAM	T	JONES	D11	942	11/04/1979
220	JENNIFER	K	LUTZ	D11	672	29/08/1968
230	JAMES	J	JEFFERSON	D21	2094	21/11/1966
240	SALVATORE	M	MARINO	D21	3780	05/12/1979
250	DANIEL	S	SMITH	D21	961	30/10/1969
260	SYBIL	P	JOHNSON	D21	8953	11/09/1975
270	MARIA	L	PEREZ	D21	9001	30/09/1980
280	ETHEL	R	SCHNEIDER	E11	8997	24/03/1967
290	JOHN	R	PARKER	E11	4502	30/05/1980
300	PHILIP	X	SMITH	E11	2095	19/06/1972
310	MAUDE	F	SETRIGHT	E11	3332	12/09/1964
320	RAMLAL	V	MEHTA	E21	9990	07/07/1965
330	WING		LEE	E21	2103	23/02/1976
340	JASON	R	GOUNOT	E21	5698	05/05/1947

Figure 4-4 Employee Table (Part 2)

Notes:

Employee Table (Part 2, Cont'd)

EMPNO	JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
180	DESIGNER	17	F	21/02/1949	21340	500	1707
190	DESIGNER	16	M	25/06/1952	20450	400	1636
200	DESIGNER	16	M	29/05/1941	27740	600	2217
210	DESIGNER	17	M	23/02/1953	18270	400	1462
220	DESIGNER	18	F	19/03/1948	29840	600	2387
230	CLERK	14	M	30/05/1935	22180	400	1774
240	CLERK	17	M	31/03/1954	28760	600	2301
250	CLERK	15	M	12/11/1939	19180	400	1534
260	CLERK	16	F	05/10/1936	17250	300	1380
270	CLERK	15	F	26/05/1953	27380	500	2190
280	OPERATOR	17	F	28/03/1936	26250	500	2100
290	OPERATOR	12	M	09/07/1946	15340	300	1227
300	OPERATOR	14	M	27/10/1936	17750	400	1420
310	OPERATOR	12	F	21/04/1931	15900	300	1272
320	FIELDREP	16	M	11/08/1932	19950	400	1596
330	FIELDREP	14	M	18/07/1941	25370	500	2030
340	FIELDREP	16	M	17/05/1926	23840	500	1907

Figure 4-5 Employee Table (Part 2, Cont'd)

Notes:

DEPARTMENT Table

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
A00	SPIFFY COMPUTER SERVICE DIV.	10	A00	
B01	PLANNING	20	A00	
C01	INFORMATION CENTER	30	A00	
D01	DEVELOPMENT CENTER	.....	A00	
D11	MANUFACTURING SYSTEMS	60	D01	
D21	ADMINISTRATION SYSTEMS	70	D01	
E01	SUPPORT SERVICES	50	A00	
E11	OPERATIONS	90	E01	
E21	SOFTWARE SUPPORT	100	E01	

Figure 4-6 DEPARTMENT Table

Notes:

DEPTNO	Department number
DEPTNAME	Department name
MGRNO	Employee serial number of the responsible manager
ADMRDEPT	Department number of the department to which this Department reports
LOCATION	Location

PROJECT Table

PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
AD3100	ADMIN SERVICES	D01	10	6.5	01/01/1982	01/02/1983	.....
AD3110	GENERAL ADMIN SYSTEMS	D21	70	6	01/01/1982	01/02/1983	AD3100
AD3111	PAYROLL PROGRAMMING	D21	230	2	01/01/1982	01/02/1983	AD3110
AD3112	PERSONNEL PROGRAMMING	D21	250	1	01/01/1982	01/02/1983	AD3110
AD3113	ACCOUNT PROGRAMMING	D21	270	2	01/01/1982	01/02/1983	AD3110
IF1000	QUERY SERVICES	C01	30	2	01/01/1982	01/02/1983	.....
IF2000	USER EDUCATION	C01	30	1	01/01/1982	01/02/1983	.....
MA2100	WELD LINE AUTOMATION	D01	10	12	01/01/1982	01/02/1983	.....
MA2110	W L PROGRAMMING	D11	60	9	01/01/1982	01/02/1983	MA2100
MA2111	W L PROGRAM DESIGN	D11	220	2	01/01/1982	01/12/1982	MA2110
MA2112	W L ROBOT DESIGN	D11	150	3	01/01/1982	01/12/1982	MA2110
MA2113	W L PROD CONT PROGS	D11	160	3	15/02/1982	01/12/1982	MA2110
OP1000	OPERATION SUPPORT	E01	50	6	01/01/1982	01/02/1983	.....
OP1010	OPERATION	E11	90	5	01/01/1982	01/02/1983	OP1000
OP2000	GEN SYSTEMS SERVICES	E01	50	5	01/01/1982	01/02/1983	.....
OP2010	SYSTEMS SUPPORT	E21	100	4	01/01/1982	01/02/1983	OP2000
OP2011	SCP SYSTEMS SUPPORT	E21	320	1	01/01/1982	01/02/1983	OP2010
OP2012	APPLICATIONS SUPPORT	E21	330	1	01/01/1982	01/02/1983	OP2010
OP2013	DB/DC SUPPORT	E21	340	1	01/01/1982	01/02/1983	OP2010
PL2100	WELD LINE PLANNING	B01	20	1	01/01/1982	15/09/1982	MA2100

Figure 4-7 PROJECT Table

Notes:

PROJNO	Project number
PROJNAME	Project name
DEPTNO	Responsible department
RESPEMP	Employee number of the responsible employee
PRSTAFF	Estimated mean staffing
PRSTDATE	Estimated starting date
PRENDATE	Estimated ending date
MAJPROJ	Major project for a subproject

Structure of an SQL Query

SELECT	- Describe the desired result Column names Arithmetic Expressions Scalar functions Column Functions
FROM	- Table names or view names
WHERE	- Conditions (Selection of rows)
GROUP BY	- Column names
HAVING	- Column names or Column sequence numbers

Figure 4-8 Structure of an SQL Query

Notes:

Clauses must be coded in the sequence indicated in the visual. Each SELECT statement must have a SELECT and a FROM clause. The other clauses are optional.

Following the SELECT keyword, you define the structure of the desired output.

The individual clauses will be discussed in detail later on in this course.

Selecting all Columns




```
SELECT *
FROM DEPARTMENT
```

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
A00	SPIFFY COMPUTER SERVICE DIV.	10	A00	
B01	PLANNING	20	A00	
C01	INFORMATION CENTER	30	A00	
D01	DEVELOPMENT CENTER	.....	A00	
D11	MANUFACTURING SYSTEMS	60	D01	
D21	ADMINISTRATION SYSTEMS	70	D01	
E01	SUPPORT SERVICES	50	A00	
E11	OPERATIONS	90	E01	
E21	SOFTWARE SUPPORT	100	E01	

Figure 4-9 Selecting all Columns

Notes:

The above statement displays all columns and all rows of the DEPARTMENT table.

The output of every SELECT statement is called the **result table.**

Attention: This is a simple statement useful for displaying small tables. In practice, you should limit the result table.

Selecting Specific Columns



For each department, I need its department number and name and the department it reports to.

```
SELECT DEPTNO,DEPTNAME,ADMRDEPT
FROM DEPARTMENT
```

DEPTNO	DEPTNAME	ADMRDEPT
A00	SPIFFY COMPUTER SERVICE DIV.	A00
B01	PLANNING	A00
C01	INFORMATION CENTER	A00
D01	DEVELOPMENT CENTER	A00
D11	MANUFACTURING SYSTEMS	D01
D21	ADMINISTRATION SYSTEMS	D01
E01	SUPPORT SERVICES	A00
E11	OPERATIONS	E01
E21	SOFTWARE SUPPORT	E01

Figure 4-10 Selecting Specific Columns

Notes:

The column names must be separated by commas. Intermediate blanks are permitted.

Selecting with Ordered Output



By the way, the listing should be ordered by the departments reported to

```
SELECT DEPTNO,DEPTNAME,ADMRDEPT
FROM DEPARTMENT
ORDER BY ADMRDEPT ASC
```

DEPTNO	DEPTNAME	ADMRDEPT
A00	SPIFFY COMPUTER SERVICE DIV.	A00
B01	PLANNING	A00
C01	INFORMATION CENTER	A00
D01	DEVELOPMENT CENTER	A00
E01	SUPPORT SERVICES	A00
D11	MANUFACTURING SYSTEMS	D01
D21	ADMINISTRATION SYSTEMS	D01
E11	OPERATIONS	E01
E21	SOFTWARE SUPPORT	E01

Figure 4-11 Selecting WITH Ordered Output

Notes:

- Resulting rows are sequenced according to columns specified by means of ORDER BY.
- Keyword ASC means sort in ascending sequence. This is the default and may be omitted.
- DESC means sort in descending sequence.
- Depending on the environment, columns used in the 'ORDER BY' clause must be a part of the result table.

Selecting with Ordered Output (Cont'd)

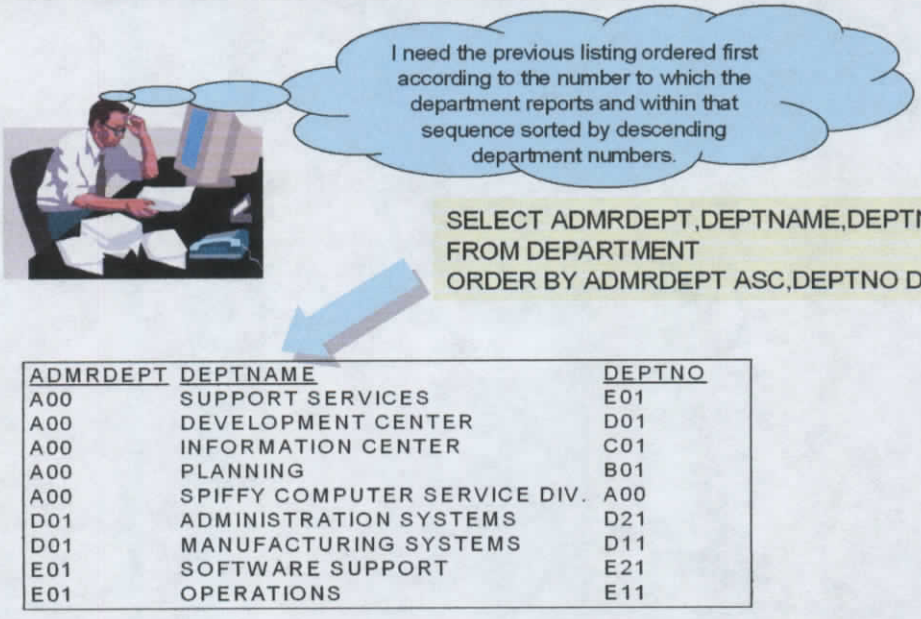


Figure 4-12 Selecting WITH Ordered Output (Cont'd)

Notes:

- If multiple columns are specified in the ORDER BY clause, the order of the columns determines their sorting priority. The first column specified defines the primary order of the rows. The later columns of the ORDER BY clause define the order of the rows for rows with the same first-column values. The rows with the same first-column values are further ordered by the values of the second column specified. The rows with identical first-and second-column values are further ordered by the values of the third column, and so on.
- ASC or DESC may be specified for each column.
- You can order date or time columns in ascending order. Assume you execute following statement:

```
SELECT EMPNO, BIRTHDATE FROM EMPLOYEE ORDER BY
BIRTHDATE DESC
```

  Then, the youngest employee (the one with the least birth date) will be listed first.
- A select column can be identified by the column name or by an integer, specifying the position in the selection list.

Alternate Specifications

```
SELECT LASTNAME, FIRSTNME, WORKDEPT,JOB,SEX
FROM EMPLOYEE
   ORDER BY WORKDEPT DESC, JOB,LASTNAME,SEX DESC

   equivalent ORDER BY clauses:
ORDER BY WORKDEPT DESC, JOB ASC,
         LASTNAME ASC, SEX DESC
ORDER BY 3 DESC,4,1,5 DESC
ORDER BY 3 DESC,4 ASC,1 ASC,5 DESC
ORDER BY WORKDEPT DESC,4 ASC, 1 ASC, SEX DESC
```

Figure 4-13 Alternate Specifications

Notes:

Only one ORDER BY clause is allowed per query.

Specify ORDER BY as the last clause in the entire SELECT statement.

Suppressing Duplicate Output Rows

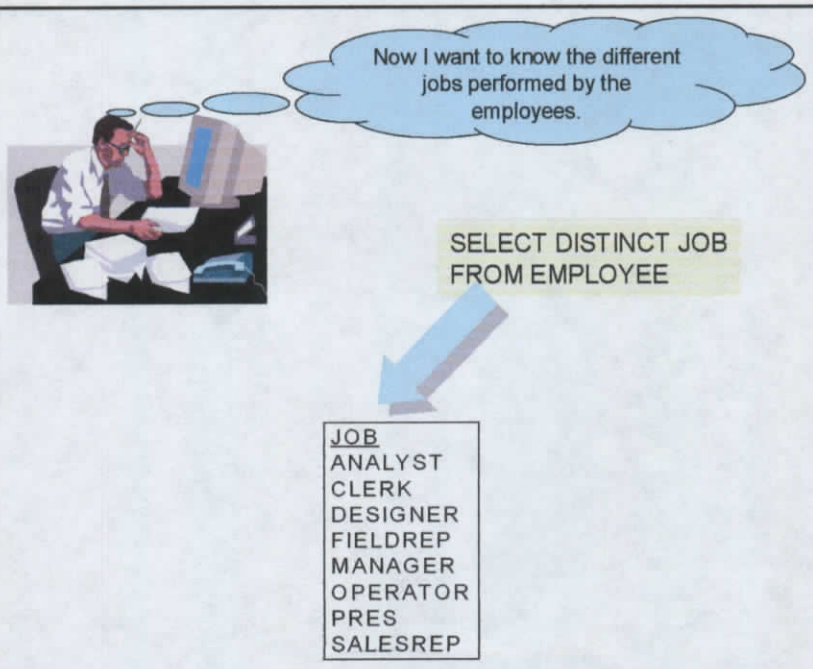
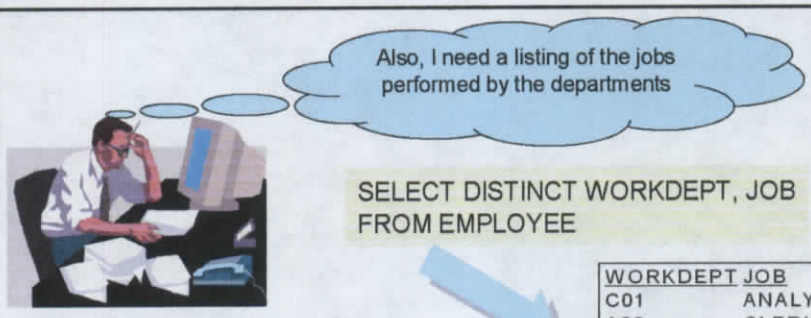


Figure 4-14 Suppressing Duplicate Output Rows

Notes:

- DISTINCT ensures that the result contains only different rows. The duplicates of these rows are removed from the result.
- The keyword DISTINCT must immediately follow the word SELECT. It is effective for all columns selected.
- The resulting rows are ordered on all columns selected in the sequence in which the columns are specified in the selection list.

Suppressing Duplicate Output Rows (Cont'd)




```
SELECT DISTINCT WORKDEPT, JOB
FROM EMPLOYEE
```

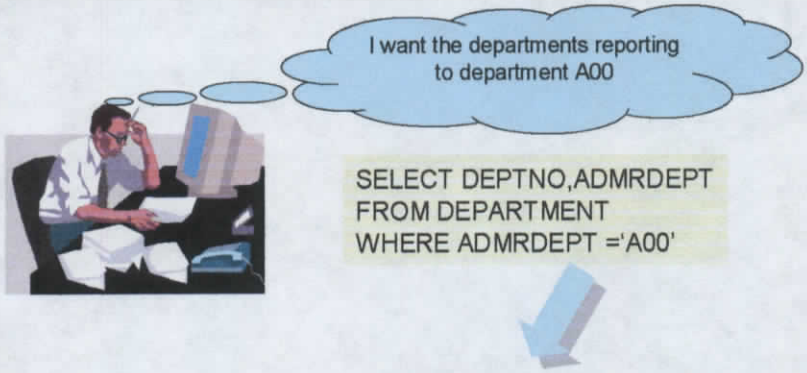
WORKDEPT	JOB
C01	ANALYST
A00	CLERK
D21	CLERK
D11	DESIGNER
E21	FIELDREP
B01	MANAGER
C01	MANAGER
D11	MANAGER
D21	MANAGER
E01	MANAGER
E11	MANAGER
E21	MANAGER
E11	OPERATOR
A00	PRES
A00	SALESREP

Figure 4-15 Suppressing Duplicate Output Rows (Cont'd)

Notes:

- DISTINCT has eliminated all rows that contain duplicate data in the set of columns specified in the SELECT statement.
- DISTINCT refers to all columns of the result table.
- DISTINCT must not be used more than once in a SELECT statement.

Selecting Rows by Character Data



DEPTNO	ADMRDEPT
A00	A00
B01	A00
C01	A00
D01	A00
E01	A00

Figure 4-16 Selecting Rows by Character Data

Notes:

To select specific rows from a table, after the SELECT statement, use the WHERE clause to specify the condition that a row must meet to be selected.

A criterion for selecting rows from a table is a search condition. A search condition consists of one or more predicates.

A predicate in the WHERE CLAUSE specifies a condition that is true, false, or unknown for a given row or group of rows. The values specified in a predicate must be compatible.

For example, if you compare a column which is defined as alphanumeric or DATE, or TIME with a value, the value to be compared must be enclosed in single quotes.

The comparison of alphanumeric columns is case sensitive.

The selection list need not encompass the columns used in the WHERE clause.

Selecting Rows by Numerical Data

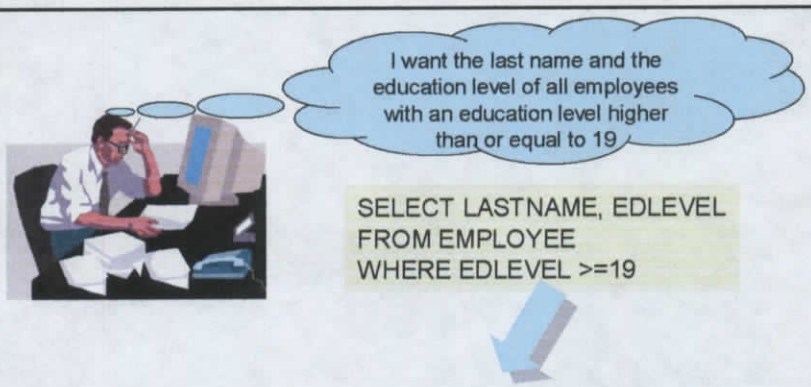


Figure 4-17 Selecting Rows by Numerical Data

Notes:

For numeric, e.g., INTEGER, SMALLINT or DECIMAL, the values to be compared are not enclosed in single quotes.

Comparison Operators

```
SELECT *
FROM EMPLOYEE
WHERE  examples:
          SALARY =  20000   equal to
          SALARY <>20000   not equal to
          SALARY >  20000   greater than
          SALARY >=20000   greater than or equal to
          SALARY <  20000   less than
          SALARY <=20000   less than or equal to
```

Figure 4-18 Comparison Operators

Notes:

The above basic predicates can be used to specify conditions in the WHERE clause.

AND and OR - Principle

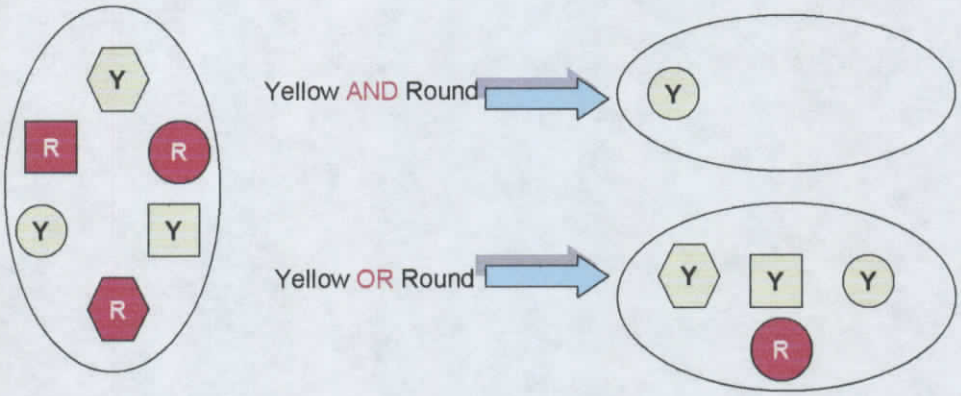


Figure 4-19 AND and OR - Principle

Notes:

When predicates are combined via AND, for a row to qualify, both predicates must be true. For predicates combined via OR, rows qualify if at least one predicate is true.

Multiple Conditions -AND

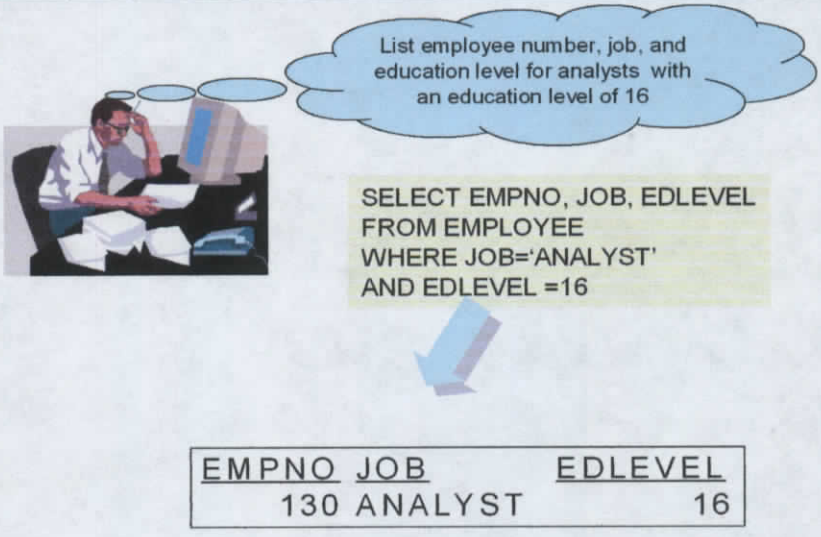


Figure 4-20 Multiple Conditions -AND

Notes:

For AND, both conditions must be met.

In the above example, only analysts with education level 16 are displayed.

Multiple Conditions -OR



List employee number, job, and education level for all analysts and all employee with an education level of 20 sorted by job and employee number

```
SELECT EMPNO, JOB, EDLEVEL
FROM EMPLOYEE
WHERE JOB='ANALYST'
OR EDLEVEL =20
ORDER BY JOB,EMPNO
```

EMPNO	JOB	EDLEVEL
130	ANALYST	16
140	ANALYST	18
30	MANAGER	20

Figure 4-21 Multiple Conditions -OR

Notes:

At least one condition must be met.

In the above example, both employees who are analysts or employees with an education level of 20 (independent of their job) are displayed.

Multiple Conditions –AND/OR



List employee number, job, and education level for all analysts with an education level of 16 and of employee with an ed-level of 18 sorted by job and employee number

```
SELECT EMPNO, JOB, EDLEVEL
FROM EMPLOYEE
WHERE JOB='ANALYST'
AND  EDLEVEL =16
OR EDLEVEL = 18
ORDER BY JOB,EMPNO
```

EMPNO	JOB	EDLEVEL
130	ANALYST	16
140	ANALYST	18
220	DESIGNER	18
20	MANAGER	18
10	PRES	18

Figure 4-22 Multiple Conditions –AND/OR

Notes:

If you combine multiple conditions by AND and OR, AND takes precedence over OR.

Multiple Conditions – Parentheses

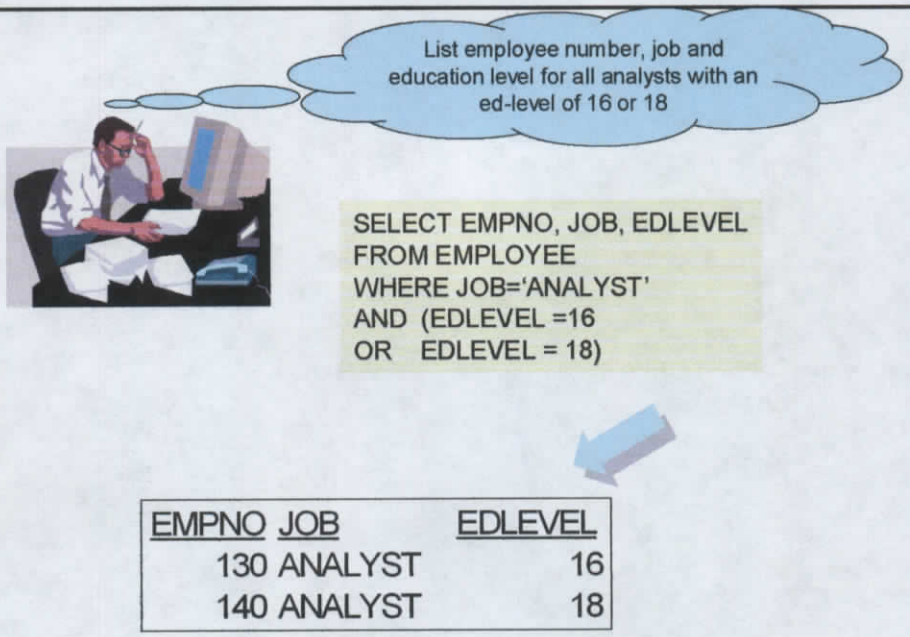


Figure 4-23 Multiple Conditions – Parentheses

Notes:

Search conditions within parentheses are evaluated first. If the order of evaluation is not determined by parentheses, AND is applied before OR. The order in which operators at the same precedence level are evaluated is undefined to allow for the optimization of search conditions.

SELECT with IN



List last name and ed-level for all employee with an ed-level of 14,19 or 20. Listing should be sorted by ed-level and last name

```
SELECT LASTNAME, EDLEVEL
FROM EMPLOYEE
WHERE EDLEVEL IN (14,19,20)
ORDER BY EDLEVEL, LASTNAME
```

EMPNO	EDLEVEL
JEFFERSON	14
LEE	14
O'CONNELL	14
SMITH	14
SPENSER	14
LUCCHESSI	19
KWAN	20

Figure 4-24 SELECT with IN

Notes:

The IN predicate compares a value with a set of values.

The values may be numeric, or alphanumeric, a date, or time.

An alphanumeric value, a date, or a time must be enclosed within single quotes.

Example:

```
SELECT *
FROM DEPARTMENT
WHERE DEPTNO IN ('B01', 'C01', 'D01')
```

SELECT with BETWEEN



List employee number and ed-level for all employees with an ed-level from 12 through 15

```
SELECT EMPNO, EDLEVEL
FROM EMPLOYEE
WHERE EDLEVEL BETWEEN 12 AND 15
ORDER BY EDLEVEL
```

EMPNO	EDLEVEL
290	12
310	12
100	14
120	14
230	14
300	14
330	14
250	15
270	15

Figure 4-25 SELECT with BETWEEN

Notes:

The BETWEEN predicate determines if a given value lies between and including two values specified in ascending order.

The clause

```
WHERE EDLEVEL BETWEEN 12 AND 15
```

Is equivalent to

```
WHERE EDLEVEL >= 12 AND EDLEVEL <= 15
```

Alphanumeric values could also be used within the between predicates. Example:

```
WHERE WORKDEPT BETWEEN 'A00' AND 'D00'
```

The following rules apply:

- Rows with the specified lower and upper limit are included in the result.
- To obtain the qualifying rows, the value preceding the AND must be lower than the value following the AND. If the limits of BETWEEN are not in ascending order, the result of the BETWEEN predicate is empty, but an error message is not returned.

Selecting NULL Values

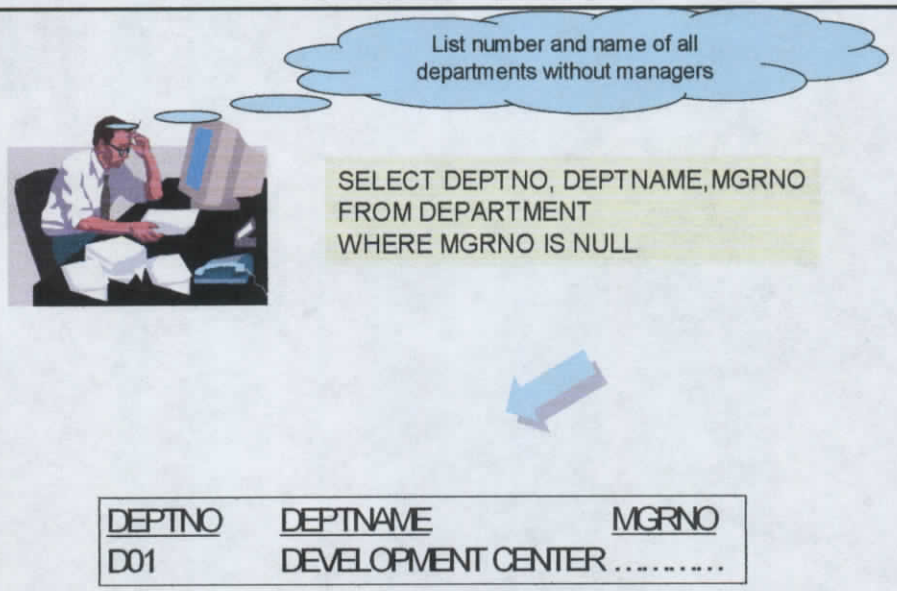


Figure 4-26 Selecting NULL Values

Notes:

The NULL predicate tests for null values.

The result of "IS NULL" cannot be unknown. If the value of the expression is null, the result is true. If the value is not null, the result is false. If "IS NOT NULL" is specified, the result is reversed.

Partial Search - LIKE

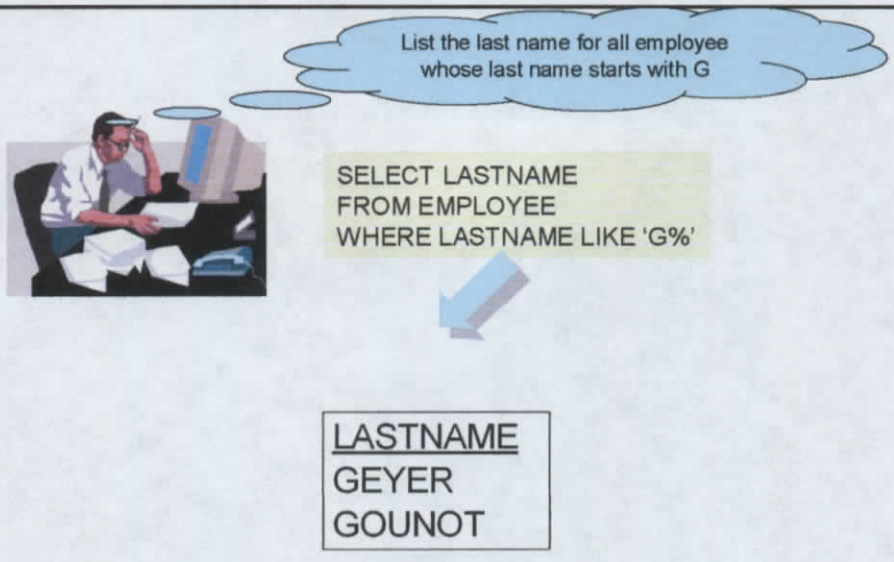


Figure 4-27 Partial Search - LIKE

Notes:

- The LIKE predicate searches for character string that have a certain pattern.
- In the pattern the percent sign ('%') serves as place holder for a string of zero or more arbitrary characters.

The above SELECT searches for names beginning with the letter 'G'.

Partial Search – Example with '%'

```
SELECT LASTNAME
FROM EMPLOYEE
WHERE LASTNAME LIKE '%SON'
```

```
THOMPSON
HENDERSON
ADAMSON
JEFFERSON
JOHNSON
```

```
SELECT LASTNAME
FROM EMPLOYEE
WHERE LASTNAME LIKE '%M%N%'
```

```
THOMPSON
ADAMSON
MARINO
```

Figure 4-28 Partial Search – Example with '%'

Notes:

The first example displays all names ending with 'SON'.

The second example displays all names containing 'M' and 'N' in the sequence. Names beginning with 'M' and ending with 'N' are displayed as well.

Partial Search – Example with '_'

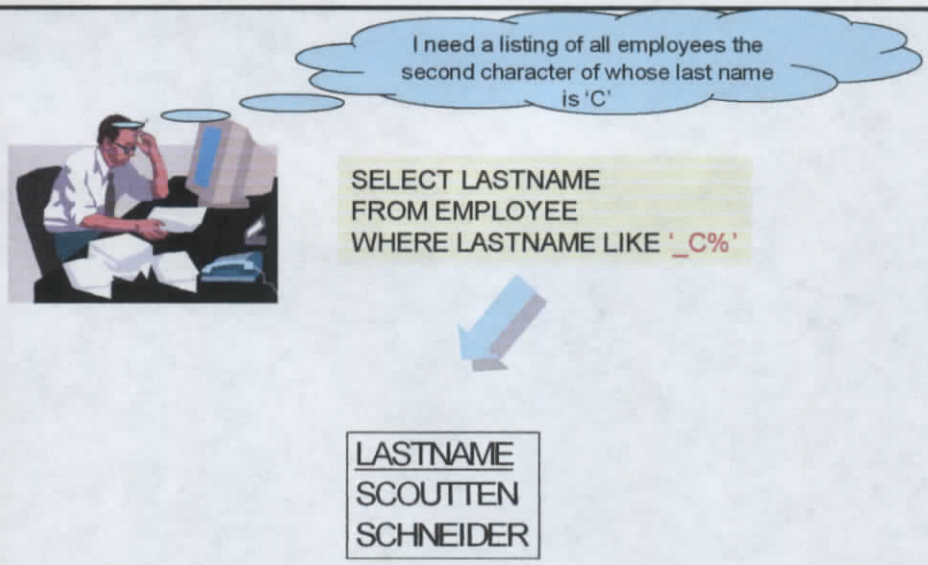


Figure 4-29 Partial Search – Example with '_'

Notes:

The underscore sign ('_') in the pattern causes the character in the corresponding position of the appropriate column of the rows not to be compared. Each underscore in the pattern represents a single character not to be compared.

The above SELECT returns names which start with an arbitrary character. The second character must be a 'C'.

If you searched for last names whose third character is a 'C', the pattern would have to start with two underscores.

Negation

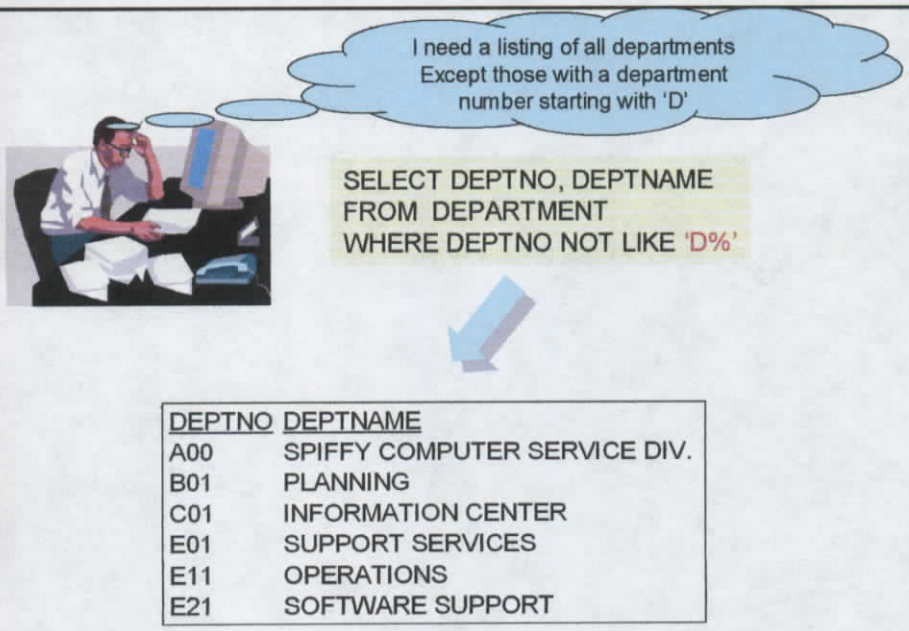


Figure 4-30 Negation

Notes:

If there are composite conditions, NOT is applied before AND and AND is applied before OR.

Summary

- Syntax of SQL SELECT statements
- Selection of entire Tables
- Ordering The result table
- Selection of specific columns
- Selection of specific rows
  - Comparison with alphanumeric columns
  - Comparison with numeric columns
  - Multiple conditions
- Use the Keywords
  - DISTINCT
  - LIKE
  - IN
  - BETWEEN

Figure 4-31 Summary

Notes:

Checkpoint

Exercise — Unit 4 Simple SQL Queries Checkpoint

1. The sequence of keywords in a SELECT statement is arbitrary.
   a. T
   b. F

2. BETWEEN causes the rows of the result table to be
   a. Sorted in ascending order
   b. Sorted in descending order
   c. Not sorted at all

3. Which SELECT keywords are optional?

Unit 5 Retrieving Data from Multiple Tables

What This Unit Is About

How to retrieve columns from more than one table or view.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Retrieve data from multiple tables.
- Know what a 'JOIN' is.
- Specify a JOIN condition.

How You Will Check Your Progress

- Checkpoint questions
- Machine labs

Objectives

After completing this unit, you should be able to:

- Retrieve data from more than one table
- Specific JOIN predicates
- Use correlation names in queries

Figure 5-1 Objectives

Notes:

5.1 Accessing Data Stored in Multiple Tables

Traditional Filing

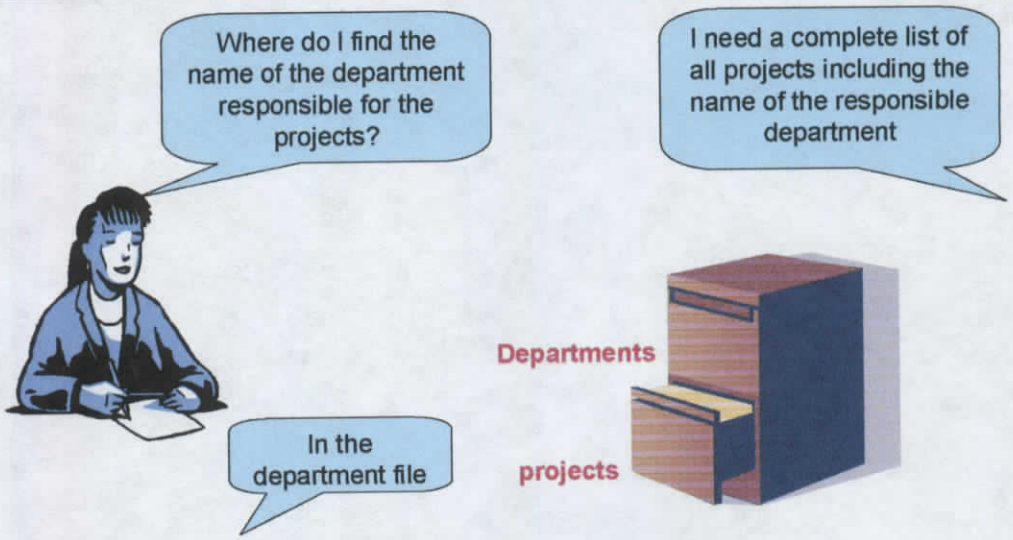


Figure 5-2 Traditional Filing

Notes:

Retrieving Data from Multiple Tables (Principle)

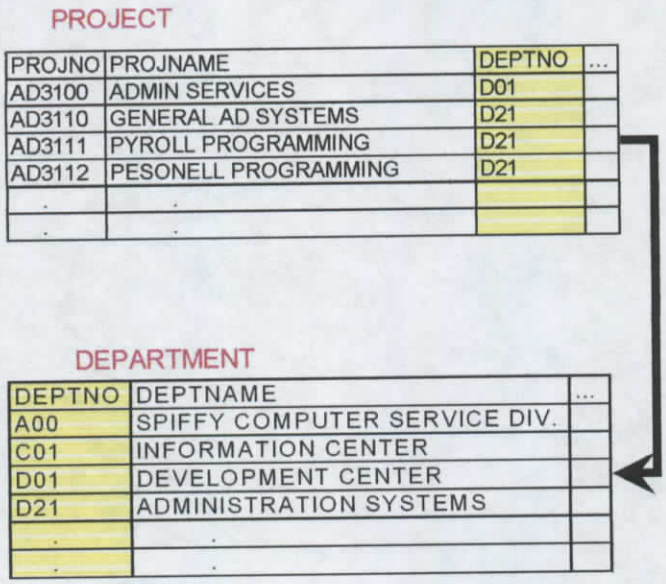


Figure 5-3 Retrieving Data from Multiple Tables (Principle)

Notes:

For each department number in table PROJECT, one row exists in table DEPARTMENT.

To find the name of the department that a project has been assigned to, DB2 might read the PROJECT table to obtain the department number for the project and then access the DEPARTMENT table using this department number for the corresponding row.

The DB2 optimizer may decide on a different access path in order to optimize performance.

Retrieving Data from Multiple Tables (JOIN)

```
SELECT PROJNO, PROJNAME,PROJECT.DEPTNO,DEPTNAME
FROM PROJECT, DEPARTMENT
WHERE PROJECT.DEPTNO=DEPARTMENT.DEPTNO
ORDER BY PROJNO
```



PROJNO	PROJNAME	DEPTNO	DEPTNAME
AD3110	GENERAL ADMIN SYSTEMS	D21	ADMINISTRATION SYSTEMS
AD3111	PAYROLL PROGRAMMING	D21	ADMINISTRATION SYSTEMS
AD3112	PERSONNEL PROGRAMMING	D21	ADMINISTRATION SYSTEMS
AD3113	ACCOUNT PROGRAMMING	D21	ADMINISTRATION SYSTEMS
...	...	...	...

Figure 5-4 Retrieving Data from Multiple Tables (JOIN)

Notes:

You can use the SELECT statement to produce reports that contain information from two or more tables. This is commonly referred as a JOIN.
To join two tables, specify the columns you want to be displayed in the SELECT clause, the table names in the FROM clause, and the join predicate in the WHERE clause.

Because DEPTNO is a column of tables DEPARTMENT and PROJECT, the column name must be qualified. Furthermore it is a good practice that all column names in a SELECT referencing more than one table be fully qualified to avoid potential error.

Join Syntax 1

```
SELECT EMPNO, LASTNAME, WORKDEPT,DEPTNAME
FROM EMPLOYEE, DEPARTMENT
WHERE WORKDEPT=DEPTNO
AND LASTNAME='HAAS'
```



```
EMPNO  LASTNAME                 WORKDEPT DEPTNAME
    10 HAAS                     A00      SPIFFY COMPUTER SERVICE DIV.
```

Figure 5-5 Join Syntax 1

Notes:

The tables EMPLOYEE and DEPARTMENT have common values for columns WORKDEPT and DEPTNO, WORKDEPT contains the department number to which the employee belongs. The join predicate tells DB2 to combine the row for an employee with the row for the department to which the employee belongs.

If you use two tables, you have to provide a single JOIN condition. If you use three tables, you must provide two JOIN conditions, and so on.

Further conditions are allowed and may be added by means of AND or OR.

Join Syntax 2 (JOIN Keyword)

```
SELECT EMPNO,LASTNAME,WORKDEPT,DEPTNAME
FROM EMPLOYEE JOIN DEPARTMENT
ON WORKDEPT=DEPTNO
WHERE LASTNAME='HAAS'
```



```
EMPNO  LASTNAME                  WORKDEPT DEPTNAME
    10 HAAS                     A00      SPIFFY COMPUTER SERVICE DIV.
```

Figure 5-6 Join Syntax 2 (JOIN Keyword)

Notes:

When the JOIN keyword is used in the FROM clause, the join predicates must be part of the ON clause.

Additional conditions must be written in the WHERE clause which must follow the ON clause.

Join with Three Tables

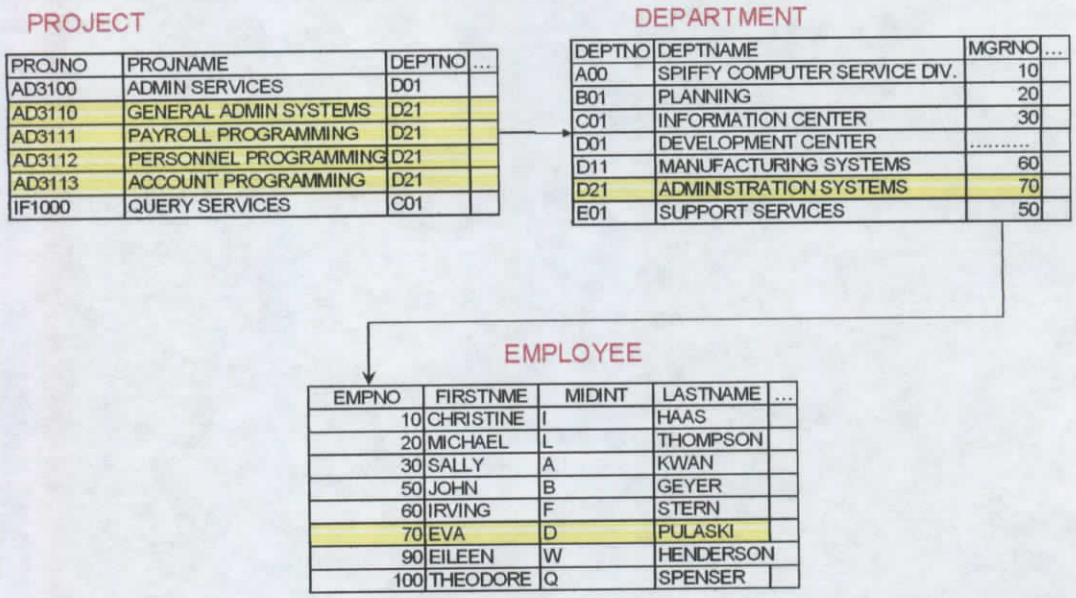


Figure 5-7 Join with Three Tables

Notes:

To find the name of the department that a project is assigned to, DB2 can first read the PROJECT table to obtain the department number of the department.

Next, read the corresponding row of the DEPARTMENT table to obtain the department name and the employee number of the responsible manager.

Next, read the corresponding row of the EMPLOYEE table to obtain the manager's last name.

The DB2 optimizer determines the sequence of the above steps.

Join with Three Tables (Cont'd)

```
SELECT PROJNO,PROJECT.DEPTNO,DEPTNAME,MGRNO,LASTNAME
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE PROJECT.DEPTNO =DEPARTMENT.DEPTNO
AND DEPARTMENT.MGRNO=EMPLOYEE.EMPNO
AND DEPARTMENT.DEPTNO='D21'
ORDER BY PROJNO
```



PROJNO	DEPTNO	DEPTNAME	MGRNO	LASTNAME
AD3110	D21	ADMINISTRATION SYSTEMS	70	PULASKI
AD3111	D21	ADMINISTRATION SYSTEMS	70	PULASKI
AD3112	D21	ADMINISTRATION SYSTEMS	70	PULASKI
AD3113	D21	ADMINISTRATION SYSTEMS	70	PULASKI

Figure 5-8 Join with Three Tables (Cont'd)

Notes:

If multiple tables are specified, but no JOIN predicate and no search condition is specified, the intermediate result table consists of all possible combinations of the rows of the specified tables (the Cartesian product).
Each row of the result is a row of the first table concatenated with a row of the second table, concatenated with a row of the third table, and so on.
The number of rows in the result is the product of the number of rows of all specified tables.

Correlation Name

```
SELECT PROJNO,P.DEPTNO,DEPTNAME,MGRNO,LASTNAME
FROM PROJECT P, DEPARTMENT D, EMPLOYEE E
WHERE P.DEPTNO=D.DEPTNO
AND D.MGRNO=E.EMPNO
AND D.DEPTNO='D21'
ORDER BY PROJNO
```



PROJNO	DEPTNO	DEPTNAME	MGRNO	LASTNAME
AD3110	D21	ADMINISTRATION SYSTEMS	70	PULASKI
AD3111	D21	ADMINISTRATION SYSTEMS	70	PULASKI
AD3112	D21	ADMINISTRATION SYSTEMS	70	PULASKI
AD3113	D21	ADMINISTRATION SYSTEMS	70	PULASKI

Figure 5-9 Correlation Name

Notes:

Correlation names can be defined in the FROM clause of a query. They may also be used in the selection list of the statement.

In this example, the correlation names are merely used as short names for the tables.

Furthermore, a correlation name can be used to avoid ambiguity or to establish a correlated reference.

Joining a Table with Itself

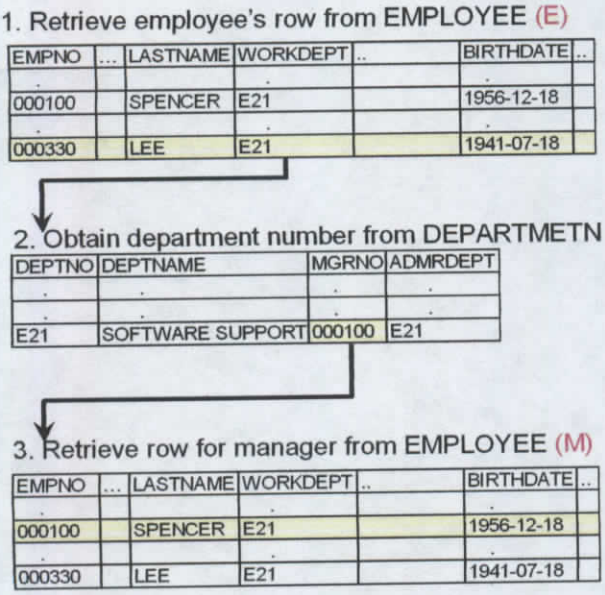


Figure 5-10 Joining a Table with Itself

Notes:

To get the required data, you have to access table EMPLOYEE twice:

For each employee using the EMPLOYEE table, you first read his/her last name, date of birth, and department number. Then, you retrieve his/her manager's employee number using the DEPARTMENT table. Finally, using the manager's employee number, you access the EMPLOYEE table again to retrieve the manager's date of birth.

As a last step, you have to compare the dates of birth.

Joining a Table with Itself (Cont'd)



Which employees are older than their manager?

```
SELECT E.EMPNO,E.LASTNAME, E.BIRTHDATE,
M.BIRTHDATE,M.EMPNO
FROM EMPLOYEE E, EMPLOYEE M,
DEPARTMENT D
WHERE E.WORKDEPT=D.DEPTNO
AND D.MGRNO=M.EMPNO
AND E.BIRTHDATE < M.BIRTHDATE
```

EMPNO	LASTNAME	BIRTHDATE	BIRTHDATE	EMPNO
000110	LUCCHESSI	1929-11-05	1933-08-24	000010
000130	QUINTANA	1925-09-15	1941-05-11	000030
000200	BROWN	1941-05-29	1945-07-07	000060
000230	JEFFERSON	1935-05-30	1953-05-26	000070
000250	SMITH	1939-11-12	1953-05-26	000070
000260	JOHNSON	1936-10-05	1953-05-26	000070
000280	SCHNEIDER	1936-03-28	1941-05-15	000090
000300	SMITH	1936-10-27	1941-05-15	000090
000310	SETRIGHT	1931-04-21	1941-05-15	000090
000320	MEHTA	1932-08-11	1956-12-18	000100
000330	LEE	1941-07-18	1956-12-18	000100
000340	GOUNOT	1926-05-17	1956-12-18	000100

Figure 5-11 Join a Table with itself (Cont'd)

Notes:

Qualifiers are required because the same table is referenced twice.

Checkpoint

Exercise — Unit 5 Multiple Tables Checkpoint

1. If you reference multiple tables in the FROM clause, you should/could use JOIN conditions to obtain the desired result.

   a. T

   b. F

2. Which of the following situations applies if you forget the JOIN conditions in a SELECT statement using multiple tables

   a. You receive an error and the statement is not executed.

   b. The statement is executed and the result is the Cartesian product of the tables.

3. Why do we use correlation names in a SELECT?

Unit 6 Scalar Functions and Arithmetic

What This Unit Is About

Scalar functions can be used to manipulate result columns. In this unit, we will discuss the format and syntax of basic scalar functions. Arithmetic allows you to present calculated values, such as salary + commission. Date columns can also be used in arithmetic calculations.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Use arithmetic to calculate values
- Specify basic scalar functions such as COALESCE/VALUE, DECIMAL, SUBSTR, and CONCAT.
- Use date and time scalar functions.
- Select different values by using CASE in a SELECT

How You Will Check Your Progress

- Checkpoint questions
- Machine labs

Objectives

After completing this unit, you should be able to :

- Use arithmetic to calculate values
- Specify basic scalar functions such as COALESCE/VALUE, DECIMAL, SUBSTR and operator CONCAT
- Use date and time scalar functions
- Select different values using by using CASE in a SELECT

Figure 6-1 Objectives

Notes:

6.1 Scalar Functions and Arithmetic

Select Calculated Values

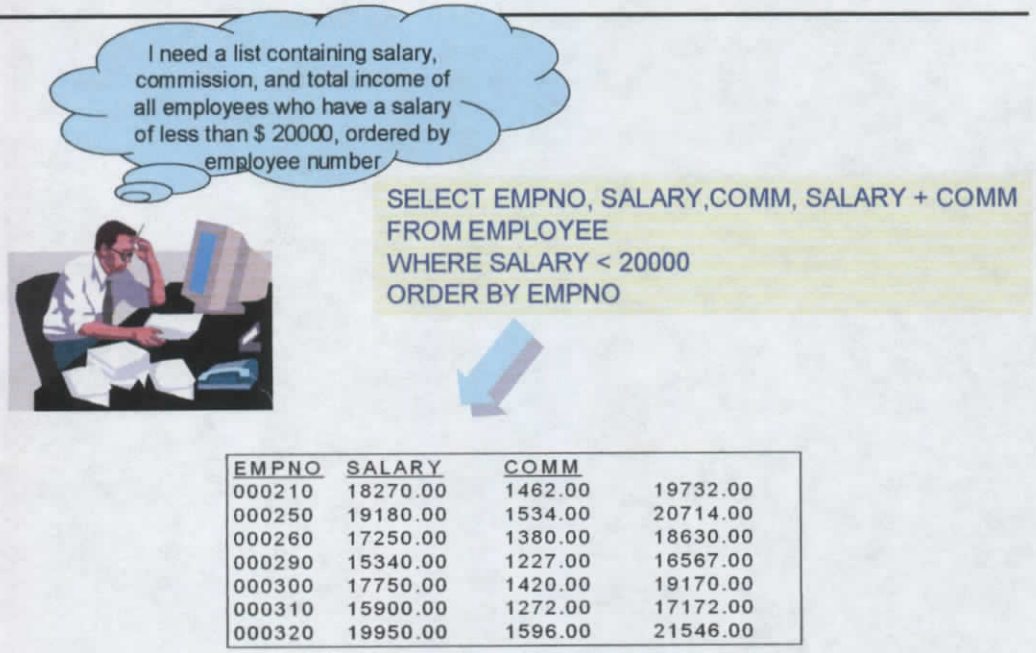


Figure 6-2 Select Calculated Values

Notes:

- You can use the following arithmetic operators and parentheses() to calculate values :

Addition	+
Subtraction	-
Multiplication	*
Division	/

- Precedence of operations:
  Expressions with parentheses are evaluated first. When the order of elevation is not specified by parentheses, prefix operators are applied before multiplication and division, and concatenation are applied before addition and subtraction. Operators at the same precedence level are applied from left to right.

- In the result table, derived columns, such as SALARY+COMM, do not have meaningful names or do not have any names (platform dependent).

Naming Result Columns

```
SELECT EMPNO, SALARY,COMM, SALARY + COMM AS INCOME
FROM EMPLOYEE
WHERE SALARY < 20000
ORDER BY EMPNO
```



EMPNO	SALARY	COMM	INCOME
000210	18270.00	1462.00	19732.00
000250	19180.00	1534.00	20714.00
000260	17250.00	1380.00	18630.00
000290	15340.00	1227.00	16567.00
000300	17750.00	1420.00	19170.00
000310	15900.00	1272.00	17172.00
000320	19950.00	1596.00	21546.00

Figure 6-3 Naming Result Columns

Notes:

Derived columns ordinarily have no or nomeaningful names dependent on the environment. Furthermore, it happens that the default name is of no particular significance to the user. In those cases, you may use the AS clause which renames the result column or assigns a name to it.

The new name can be used in the ORDER BY clause but not in the WHERE clause.

Substitution of NULL Values

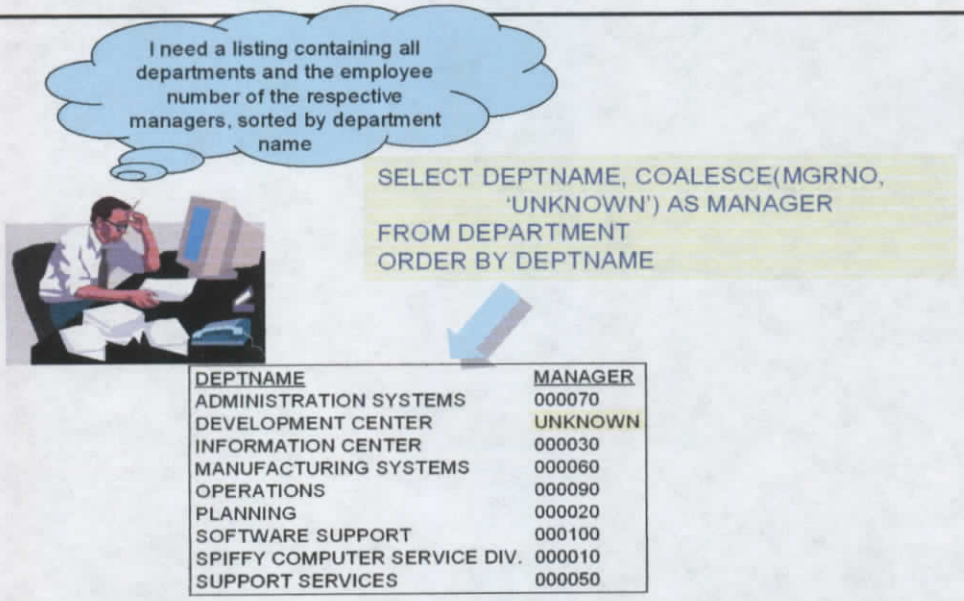


Figure 6-4 Substitution of NULL Values

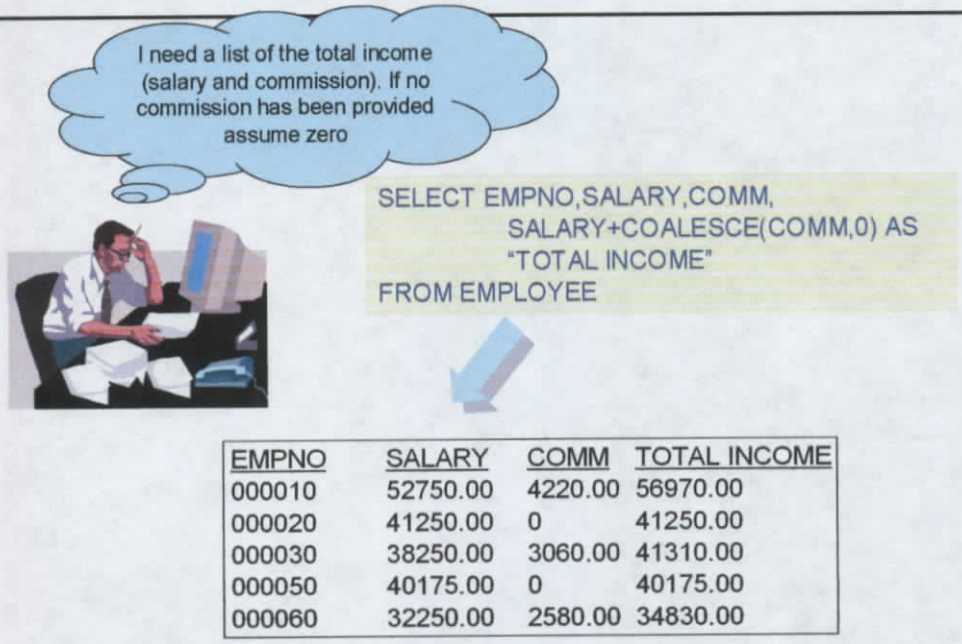
Notes:

- The scalar function COALESCE is equivalent to VALUE.
- COALESCE allows multiple parameters:
- COALESCE(expression,expression,'constant').

- The COALESCE function takes a variable number of parameters and returns the value of the first parameter that is not null. The result is null if the values of all parameters are null.

- In the example, the value "UNKNOWN" is returned if a department does not have a manager, i.e., MGRNO is NULL.

Arithmetic with NULL values



EMPNO	SALARY	COMM	TOTAL INCOME
000010	52750.00	4220.00	56970.00
000020	41250.00	0	41250.00
000030	38250.00	3060.00	41310.00
000050	40175.00	0	40175.00
000060	32250.00	2580.00	34830.00

Figure 6-5 Arithmetic with NULL values

Notes:

If one of the columns used in the expression contains a NULL value, the result of the expression is unknown.

In the example, if commission is NULL, zero is added to the salary.

Calculated Values

```
SELECT EMPNO,SALARY,SALARY*1.0375
FROM EMPLOYEE
WHERE SALARY <20000
ORDER BY EMPNO
```



EMPNO	SALARY	
000210	18270.00	18955.125000
000250	19180.00	19899.250000
000260	17250.00	17896.875000
000290	15340.00	15915.250000
000300	17750.00	18415.625000
000310	15900.00	16496.250000
000320	19950.00	20698.125000

Figure 6-6 Calculated Values

Notes:

The above example shows multiplication using a constant. You can also use calculations with multiple columns (e.g., COL1 * COL2).

Decimal Representation of a Value

```
SELECT EMPNO, SALARY,
DECIMAL(SALARY*1.0375,8,2)
FROM EMPLOYEE
WHERE SALARY <20000
ORDER BY EMPNO
```



EMPNO	SALARY	
000210	18270.00	18955.12
000250	19180.00	19899.25
000260	17250.00	17896.87
000290	15340.00	15915.25
000300	17750.00	18415.62
000310	15900.00	16496.25
000320	19950.00	20698.12

Figure 6-7 Decimal Representation of a Value

Notes:

The DECIMAL function returns a decimal representation of a numeric value or of a character value representing a number. In the example above, the DECIMAL function is used to ensure that the result column has 2 decimal positions.

The result is truncated after the second decimal position.

Decimal Values – Truncation and Rounding

```
SELECT EMPNO, SALARY,
DECIMAL(SALARY*1.0375+0.005,8,2)
FROM EMPLOYEE
WHERE SALARY <20000
ORDER BY EMPNO
```



EMPNO	SALARY	
000210	18270.00	18955.13
000250	19180.00	19899.25
000260	17250.00	17896.88
000290	15340.00	15915.25
000300	17750.00	18415.63
000310	15900.00	16496.25
000320	19950.00	20698.13

Figure 6-8 Decimal Values – Truncation and Rounding

Notes:

If you need a rounded result, add 5 to the first decimal position to be truncated and apply the DECIMAL function to this expression.

Some environments support a ROUND function solving this problem.

Condition on Calculated Values

```
SELECT EMPNO,COMM,SALARY,(COMM/SALARY)*100
FROM EMPLPYEE
WHERE (COMM/SALARY)*100>8
ORDER BY EMPNO
```



EMPNO	COMM	SALARY	
000140	2274.00	28420.00	8.001400
000210	1462.00	18270.00	8.002100
000240	2301.00	28760.00	8.000600
000330	2030.00	25370.00	8.001500

Figure 6-9 Condition on Calculated Values

Notes:

Arithmetic expressions can be used in the SELECT list and in the WHERE condition.

The SELECT only lists rows for employees with a commission higher than 8 percent of their salary.

Date and Time

- DATE, TIME, TIMESTAMP data internally sorted as packed decimal, without sign

Data Type	Internal Format	Internal Length
DATE	yyyymmdd	4 bytes
TIME	hhmmss	3 bytes
TIMESTAMP	yyyymmddhhmmssnnnnnn	10 bytes

- Program uses an external format

Fortmat	Timeformat	Length	Date Format	Length
ISO	hh.mm.ss	8 bytes	yyyy-mm-dd	10 bytes
USA	hh:mm AM / hh:mm PM	8 bytes	mm/dd/yyyy	10 bytes
EUR	hh.mm.ss		dd.mm.yyyy	10 bytes
JIS	hh:mm:ss	8 bytes	yyyy-mm-dd	10 bytes
LOCAL	???	???	???	???
TIMESTAMP DATA yyyy-mm-dd-hh.mm.ss.nnnnnn				26 bytes

Figure 6-10 Date and Time

Notes:

The visual shows the internal and external formats for DATE/TIME data.

A timestamp is a seven-part value (year, month, day, hour, minute, second, and microsecond) that represents a date and a time including microseconds.

Comparison with Dates

```
SELECT EMPNO,LASTNAME,BIRTHDATE
FROM EMPLOYEE
WHERE BIRTHDATE>='1955-01-01'
ORDER BY BIRTHDATE
```



EMPNO	LASTNAME	BIRTHDATE
000160	PIANKA	1955-04-12
000100	SPENSER	1956-12-18

Figure 6-11 Comparison with Dates

Notes:

Columns defined as DATE or TIME can be compared with other DATE or TIME columns or with a valid date or time.

Constant values must be placed in single quotes. All legal formats can be used.

In the shown example, all employees are selected that were born '1955-01-01' or later.

DATE/TIME Arithmetic

- Subtraction Only
  time – time = time duration (decimal(6,0))
  date – date = date duration (decimal(8,0))
  timestamp – timestamp = timestamp duration (decimal (20,6))

- Labeled durations :YEAR, MONTHS, DAYS, HOURS, MINUTES,SECONDS, MICROSECONDS
  time +- duration =time
  date +- duration = date
  timestamp +- duration =timestamp

Figure 6-12 DATE/TIME Arithmetic

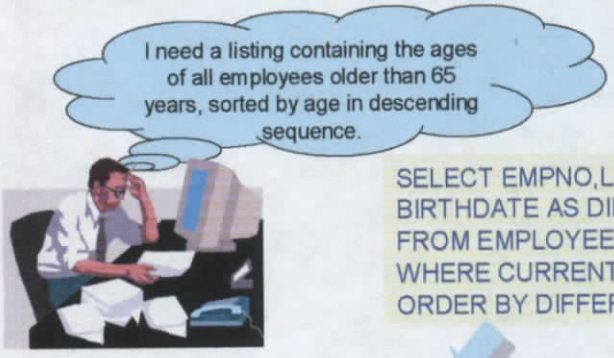
Notes:

A date duration represents the period between two dates. The result is presented in format yyyymmdd.

A time duration represents the period between two times. The result is presented in format hhmmss.

A timestamp duration represent the period between two timestamps. The result is presented in format yyyymmddhhmmss.nnnnnn (the nnnnnn portion represents microseconds).

Subtraction of dates




```
SELECT EMPNO,LASTNAME,CURRENT_DATE-
BIRTHDATE AS DIFFER
FROM EMPLOYEE
WHERE CURRENT_DATE-BIRTHDATE >650000
ORDER BY DIFFER DESC
```

EMPNO	LASTNAME	DIFFER
000130	QUINTANA	721116
000050	GEYER	721116
000340	GOUNOT	720314
000110	LUCCHESI	680926
000310	SETRIGHT	670410
000320	MEHTA	660020

Figure 6-13 Subtraction of dates

Notes:

In above example, the values of two date columns are subtracted. The result of the subtraction is presented in the format yyyymmdd. Leading zeros are not displayed.

For example, in the first result line, the value 721116 means 72 years, 11 months, and 16 days.

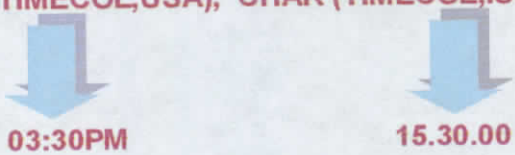
In the WHERE clause, the calculated durations are compared with 650000,i.e., 65 years, 00months, and 00 days (older than 65 years).

CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP are special registers, provided by DB2, containing the actual date, actual time, or actual timestamp.

Date / Time Scalar Functions

- CHAR controls external format of date / time data

SELECT CHAR **(TIMECOL,USA), CHAR (TIMECOL,ISO)...**



- DAY,MONTH,YEAR,HOUR,MINUTE,SECOND,MICROSECOND, DATE,TIME
  Extract portions of a date, time, timestamp, or duration
- DAYS
  Converts date in number of days

Figure 6-14 Date / Time Scalar Functions

Notes:

The visual shows the scalar functions for date and time values.

Date Scalar Functions

```
SELECT LASTNAME,FIRSTNAME, CURRENT_DATE-BIRTHDATE AS AGE,
YEAR(CURRENT_DATE-BIRTH_DATE)AS YEAR,
MONTH(CURRENT_DATE-BIRTHDATE)AS MONTH,
DAY(CURRENT_DATE-BIRTHDATE)AS DAYS
FROM EMPLOYEE
WHERE YEAR(CURRENT_DATE-BIRTHDATE)>65
ORDER BY AGE DESC, LASTNAME
```



LASTNAME	FIRSTNAME	AGE	YEARS	MONTHS	DAYS
GEYER	JHON	721116	72	11	16
QUINTANA	DOLORES	721116	72	11	16
GOUNOT	JASON	720315	72	3	15
LUCCHES	VINCENZO	680926	68	9	26
ISETRIGHT	MAUDE	670410	67	4	10
MEHTA	RAMLAL	660020	66	0	21

Figure 6-15 Date Scalar Functions

Notes:

In the above example, year, month, and day are displayed separately using appropriate scalar functions.

The resulting columns normally do not have column heading, the AS clause is used to name the columns.

DATE Arithmetic

```
SELECT PROJNO, DAYS (PRENDATE)-DAYS(PRSTDATE) AS DAYS
FROM PROJECT
WHERE DAYS(PRENDATE)-DAYS(PRSTDATE)<=300
ORDER BY DAYS
```



PROJNO	DAYS
PL2100	257
MA2113	289

Figure 6-16 DATE Arithmetic

Notes:

The example shows the duration of projects in days. To obtain a positive result, the project start date (PRSTDATE) must be subtracted from the project end date (PRENDATE) because the end date is higher.

DATE Arithmetic (Cont'd)

```
SELECT PROJNO,PRENDATE,PRENDATE+2MONTHS + 15 DAYS
FROM PROJECT
WHERE PROJNO='AD3100'
ORDER BY PROJNO
```



PROJNO	PRENDATE	
AD3100	1983-02-01	1983-04-16

Figure 6-17 DATE Arithmetic (Cont'd)

Notes:

The example shows the use of labeled durations and adds 2 months and 15 days to a date.

Substring of Strings

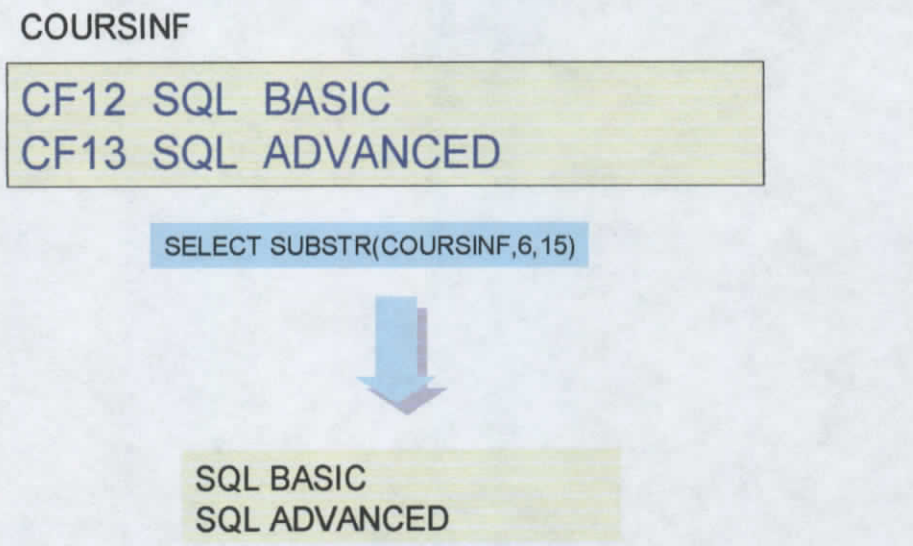


Figure 6-18 Substring of Strings

Notes:

If only a part of a character string is to be selected, use the SUBSTR scalar function to extract the part.

Assume that COURSINF is defined CHAR(20) and that both course code and course title are stored in that column, separated by a blank. To only display the course title, you can use the SUBSTR scalar function.

SUBSTR(COURSINF,6,15) returns a substring of length 15 beginning at position 6.i.e., the course title in the above example.

Substring of Strings (Cont'd)

```
SELECT SUBSTR(PROJNO,1,2)AS PROJ_CLASS,
       PROJNAME
FROM PROJECT
WHERE PROJNO LIKE 'IF%'
```



PROJ_CLASS	PROJNAME
IF	QUERY SERVECES
IF	USER EDUCATION

Figure 6-19 Substring of Strings (Cont'd)

Notes:

The first two characters of the project number function represent the project class. To only display the project class, the SUBSTR function is used in the above example.

SUBSTR(PROJNO,1,2) extracts the first two characters.

Concatenation of Values

```
SELECT LASTNAME CONCAT ','
          CONCAT FIRSTNME AS NAME
FROM EMPLOYEE
WHERE WORKDEPT ='A00'
ORDER BY NAME
```



NAME
HAAS,CHRISTA
LUCCHESI, VICENZO
O'CONNELL,SEAN

Figure 6-20 Substring of Strings (Cont'd)

Notes:

The above example displays last name and first name as a single column, separated by a comma.

CONCAT combines character string to a single column.

Double bars (||) are equivalent to the CONCAT operator.

Examples:

```
SELECT LASTNAME || ',' || ',' FIRSTNME FROM EMPLOYEE
```

The CONCAT operator is more independent of environments, such as keyboards.

Case Expression

```
SELECT EMPNO,FIRSTNME,MIDINIT,LASTNAME,
        CASE
        WHEN EDLEVEL <15 THEN 'SECONDARY'
        WHEN EDLEVEL <19 THEN 'COLLEGE'
        ELSE 'POST GRADUATE'
        END
        AS EDUCATION
FROM EMPLOYEE
```

Figure 6-21 Case Expression

Notes:

CASE allows different expressions to be selected for a result column based on the evaluation of one or more conditions.

Depending on the contents of the EDLEVEL column, different character strings are displayed in the output listing.

Case Expression (Cont'd)

```
SELECT EMPNO,FIRSTNME,LASTNAME,
        CASE SEX
        WHEN 'F' THEN FEMALE
        WHEN 'M' THEN 'MALE'
        ELSE 'INVALED VALUE'
        END AS SEX
FROM EMPLOYEE
WHERE SALARY <18000
```



EMPNO	FIRSTNME	LASTNAME	SEX
000260	SYBIL	JOHNSON	FEMALE
000290	JOHN	PARKER	MALE
000300	PHILIP	SMITH	MALE
000310	MAUDE	SETRIGHT	FEMALE

Figure 6-22 Case Expression (Cont'd)

Notes:

One purpose of CASE expression is to replace the values in a result table with more meaningful values.

In the EMPLOYEE table, the sex of an employee is stored as a single character. The stored values are F and M. Using CASE in above example offers the possibility to display the more meaningful texts FEMALE instead of F and MALE instead of M.

This second example shows a simple when-clause. This means that the value of the expression following the CASE keyword is evaluated against the values of the WHEN expressions.

Summary

- Arithmetic can be used in the SELECT list and in the WHERE clause
- Columns may be named by using the AS clause
- Scalar functions can be used to manipulated the data representation (DECIMAL,CONCAT,SUBSTR)
- DB2 supports arithmetic with date, time or timestamp data
- DB2 supports ordering of date or time columns
- Scalar functions can be used to extract hour, minute, or second of a time or of a time difference
- CASE expression allows an expression to be selected based the evaluation of one or more conditions

Figure 6-23 Summary

Notes:

Checkpoint

Exercise — Unit 6 Scalar Functions and Arithmetic Checkpoint

1. If you use the DECIMAL scalar function to have two decimal positions instead of six, the result will be rounded.

   a. T

   b. F

2. If you subtract two dates, the format of the result will be:

   a. DD.MM.YYYY

   b. YYYY-MM-DD

   c. YYYYMMDD

3. Names four scalar functions.

   _____

   _____

   _____

   _____

Unit 7 Column Functions and Grouping

What This Unit Is About

This unit introduces column functions which can be used to obtain result derived from one or more rows. In addition, the syntax for selecting group results will be taught.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Know the difference between scalar and column functions.
- Know which column functions are supported by DB2.
- Use column functions to group rows.

How You Will Check Your Progress

- Checkpoint questions
- Machine labs

Objectives

After completing this unit, you should be able to :

- Know the difference between scalar and column functions
- Know which column functions are supported by DB2
- Group individual rows into combined sets based on one or more columns

Figure 7-1 Objectives

Notes:

7.1 Column Functions and Grouping

Column Functions

• Compute Total	**SUM (expression)**
• Calculate Average	**AVG (expression)**
• Find Minimum Value	**MIN (expression)**
• Find Maximum Value	**MAX (expression)**
• Determine the number of meeting the search condition	**COUNT(*)**
• Determine the number of unique values in a specified column – NULL value not counted	**COUNT(distinct column name)**

Figure 7-2 Column Functions

Notes:

For SUM, AVG, MIN, MAX, COUNT(*), COUNT(DISTINCT ...), the following applies:

- The bases for these column functions are values derived from one or more rows.
- The data type of the input values for SUM and AVG must be numeric.

Example for COUNT(*):

```
SELECT COUNT(*) FROM EMPLOYEE WHERE WORKDEPT = 'A00'
```

Returns the number of **rows** having the value 'A00' in the WORKDEPT column. In general, the argument for COUNT(*) is a set of rows. The result is the number of rows in the set. Any row that includes only null values is includes in the count.

Example for COUNT(DISTINCT column-name):

```
SELECT COUNT(DISTINCT WORKDEPT) FROM EMPLOYEE
```

Returns the number of **unique values** in the WORKDEPT column. Null values are not counted. Furthermore, the argument of this function can be an expression instead of a column name.

Sample Column Functions

```
SELECT   SUM(SALARY) AS SUM,
         AVG(SALARY) AS AVG,
         MIN(SALARY) AS MIN,
         MAX(SALARY) AS MAX,
         COUNT(*)    AS COUNT,
         COUNT(DISTINCT WORKDEPT) AS DEPT
FROM EMPLOYEE
```



SUM	AVG	MIN	MAX	COUNT	DEPT
873715.00	27303.59375000	15340.00	52750.00	32	8

Figure 7-3 Sample Column Functions

Notes:

The basis for column functions are all rows of the employee table.

The result does not contain detailed information for individual rows.

Precision and scale of the resulting values are derived from the definition for the column to which the column function applies.

Except for COUNT(*), expressions (e.g., SALARY + COMM) may be used as an argument for column functions.

Column Functions Based on Subset

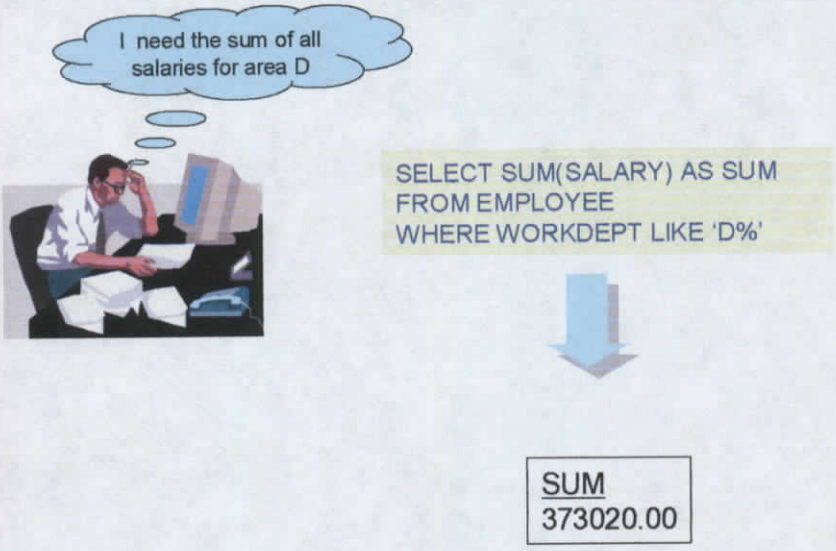
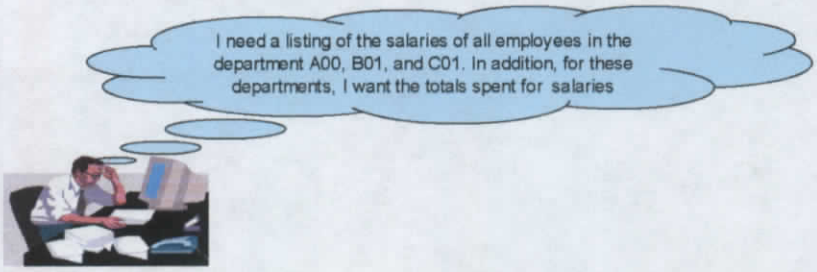


Figure 7-4 Column Functions Based on Subset

Notes:

The result is derived only from rows for which the predicate of the WHERE clause is true.

GROUP BY




```
SELECT WORKDEPT, SALARY
FROM EMPLOYEE
WHERE WORKDEPT IN('A00','B01','C01')
ORDER BY WORKDEPT
```

```
SELECT WORKDEPT, SUM(SALARY) AS SUM
FROM EMPLOYEE
WHERE WORKDEPT IN('A00','B01','C01')
GROUP BY WORKDEPT
ORDER BY WORKDEPT
```

WORKDEPT	SALARY
A00	52750.00
A00	46500.00
A00	29250.00
B01	41250.00
C01	38250.00
C01	23800.00
C01	28420.00

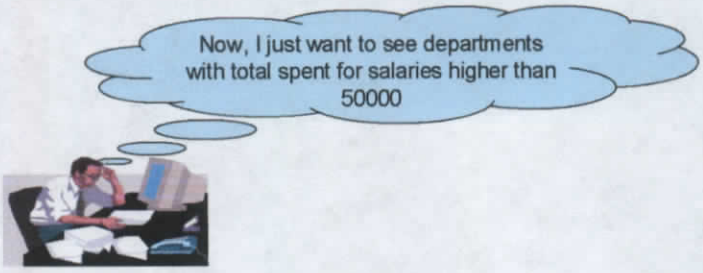
WORKDEPT	SALARY
A00	128500.00
B01	41250.00
C01	90470.00

Figure 7-5 Column Functions Based on Subset

Notes:

- With GROUP BY, a column functions result in a single value for **each group.**
- GROUP BY can only be used in conjunction with column functions.
- All nulls are a single group.
- A name defined by an AS clause cannot be used in GROUP BY.
- GROUP BY **does not guarantee a sorted result table.** If an ordered result table is required an appropriate ORDER BY clause must be specified.

GROUP BY - HAVING




```
SELECT WORKDEPT, SUM(SALARY) AS SUM
FROM EMPLOYEE
WHERE WORKDEPT IN('A00','B01','C01')
GROUP BY WORKDEPT
ORDER BY WORKDEPT
```

```
SELECT WORKDEPT, SUM(SALARY) AS SUM
FROM EMPLOYEE
WHERE WORKDEPT IN('A00','B01','C01')
GROUP BY WORKDEPT
HAVING SUM(SALARY)>50000
ORDER BY WORKDEPT
```

WORKDEPT	SALARY
A00	128500.00
B01	41250.00
C01	90470.00

WORKDEPT	SALARY
A00	128500.00
C01	90470.00

Figure 7-6 GROUP BY - HAVING

Notes:

So far we built groups. With HAVING, we now can select groups satisfying certain conditions. Conditions for groups must be specified in the having clause.

HAVING may specify any column function on any column in a table being queried. This column need not be in the SELECT list.

HAVING can only be used in conjunction with GROUP BY.

GROUP BY – HAVING (Cont'd)

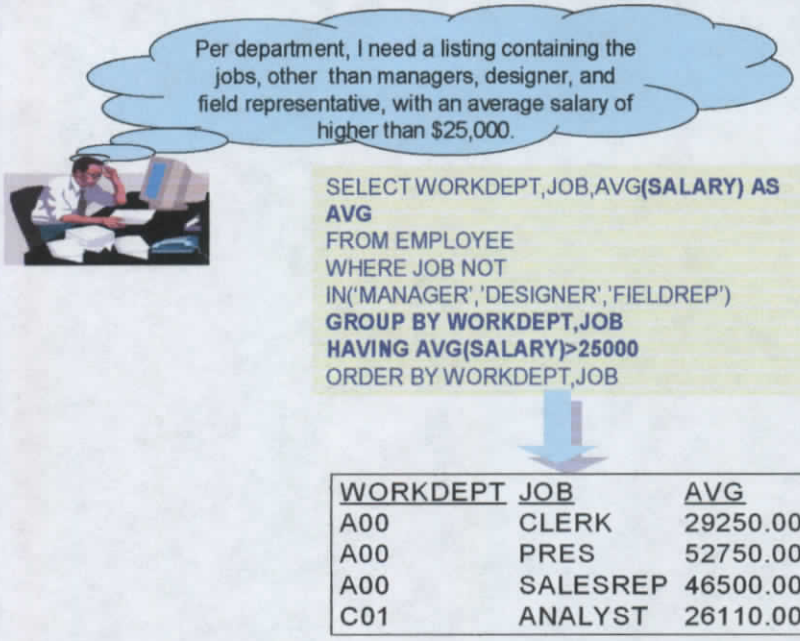


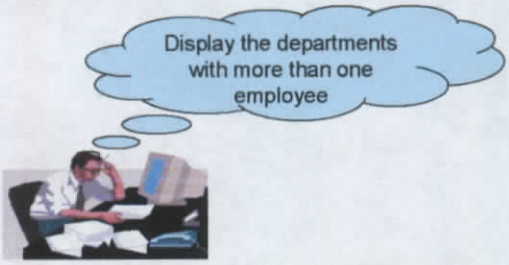
Figure 7-7 GROUP BY – HAVING (Cont'd)

Notes:

In the above example, a GROUP BY clause with two columns is shown.

The first three rows in the result have the same value ('A00') in the WORKDEPT column, but different values in the JOB column. The result of the group by clause are three different groups for department 'A00'.
Only rows with the same values in columns WORKDEPT and JOB will be grouped.

Examples with HAVING




```
SELECT WORKDEPT COUNT(*) AS NUMB
FROM EMPLOYEE
GROUP BY WORKDEPT
ORDER BY NUMB,WORKDEPT
```

WORKDEPT	NUMB
A00	1
E01	1
A00	3
C01	3
E21	4
E11	5
D21	6
D11	9

```
SELECT WORKDEPT COUNT(*) AS NUMB
FROM EMPLOYEE
GROUP BY WORKDEPT
HAVING COUNT(*)>1
ORDER BY NUMB,WORKDEPT
```

WORKDEPT	NUMB
A00	3
C01	3
E21	4
E11	5
D21	6
D11	9

Figure 7-8 Examples with HAVING

Notes:

SELECT 1 is displayed to show the result without a HAVING clause (step1 of SELECT 2).

SELECT 2: Determines the result table consisting of groups (see SELECT 1) for which the search condition of the HAVING clause is true.

The condition COUNT(*) > 1 applies to the result table of SELECT 1.

Examples with HAVING (Cont'd)




```
SELECT WORKDEPT, AVG(EDLEVEL) AS ED,
   AVG(YEAR(CURRENT_DATE-HIREDATE))
   AS YEAR
FROM EMPLOYEE
GROUP BY WORKDEPT
ORDER BY 2
```

WORKDEPT	ED	YEARS
E11	14	27
E21	15	31
D21	15	22
E01	16	49
D11	16	24
A00	17	35
B01	18	24
C01	18	23

```
SELECT WORKDEPT, AVG(EDLEVEL) AS ED,
   AVG(YEAR(CURRENT_DATE-HIREDATE))
   AS YEAR
FROM EMPLOYEE
GROUP BY WORKDEPT
HAVING AVG(YEAR(CURRENT_DATE-HIREDATE))>=30
ORDER BY 2
```

E21	15	31
E01	16	49
A00	17	35

Figure 7-9 Examples with HAVING (Cont'd)

Notes:

Question (SELECT 1):

Question (SELECT 2):

Examples with HAVING (Cont'd)




```
SELECT WORKDEPT, AVG(EDLEVEL)AS
ED,MIN(BONUS)AS MIN
FROM EMPLOYEE
GROUP BY WORKDEPT
```

WORKDEPT	ED	MIN
A00	17	600.00
B01	18	800.00
C01	18	500.00
D11	16	400.00
D21	15	300.00
E01	16	800.00
E11	14	300.00
E21	15	400.00

```
SELECT WORKDEPT, AVG(EDLEVEL)AS
ED,MIN(BONUS)AS MIN
FROM EMPLOYEE
GROUP BY WORKDEPT
HAVING MIN(BONUS)=300
ORDER BY 2
```

WORKDEPT	ED	MIN
D21	15	300.00
E11	14	300.00

Figure 7-10 Examples with HAVING (Cont'd)

Notes:

Question:

Question:

Summary

- Column functions may be specified only in
  - **SELECT**
  - **HAVING**
- **SELECT** may specify only
  - Column functions
  - Columns specified in **'GROUP BY'**
- **HAVING** may specified
  - Any column function on any column in a table being queried. This column need not be in **SELECT**
- Column functions may not be nested

Figure 7-11 Summary

Notes:

Checkpoint

Exercise — Unit 7 Column Functions and Grouping Checkpoint

1. NULL cannot be a group.

    a. T

    b. F

2. If you use GROUP BY, you must specify a column function in the select list.

    a. T

    b. F

3. Which of the following statements is correct:

    a. GROUP BY does not guarantee a stored result table.

    b. Column functions can be used in a where clause.

    c. COUNT(*) counts columns of a table.

Unit 8 UNION

What This Unit Is About

This unit describes the syntax of SQL for obtaining the combined result of two or more SELECTs.

What You Should Be Able to Do

After completing this unit, you should be able to:

Combine the results of several queries in a single result.

How You Will Check Your Progress

- Checkpoint questions
- Machine labs

Objectives

After completing this unit, you should be able to :

- Derive a result by combining two or more results
- Know the syntax rules for UNION statement
- Know the rules for the select list
- Know the difference between UNION and UNION ALL

Figure 8-1 Objectives

Notes:

8.1 UNION

UNION - Principle

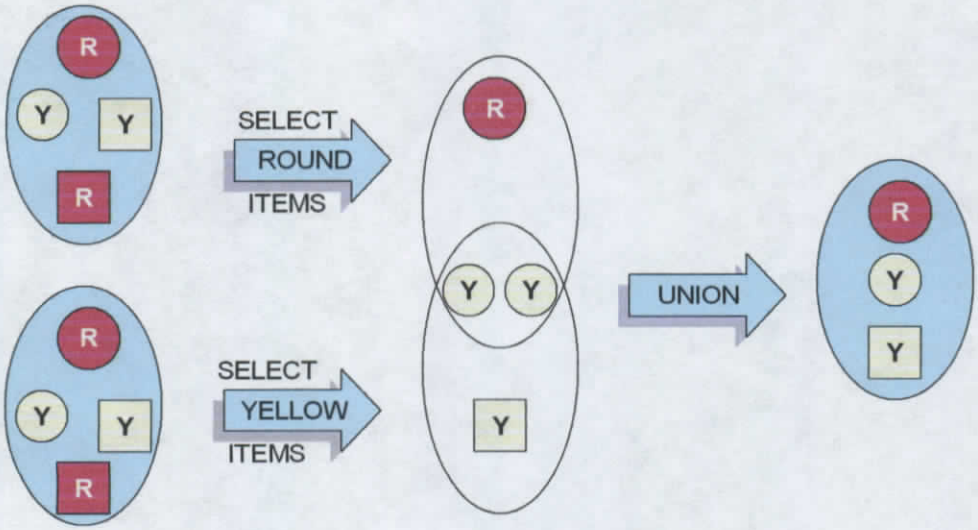


Figure 8-2 UNION - Principle

Notes:

Each oval means a set of items (red and yellow squares and circles).

SQL is based on mathematically defined set theory.

According to set theory, the combination of two sets results in a new set containing each different item only once.

Combination of Multiple Result Tables

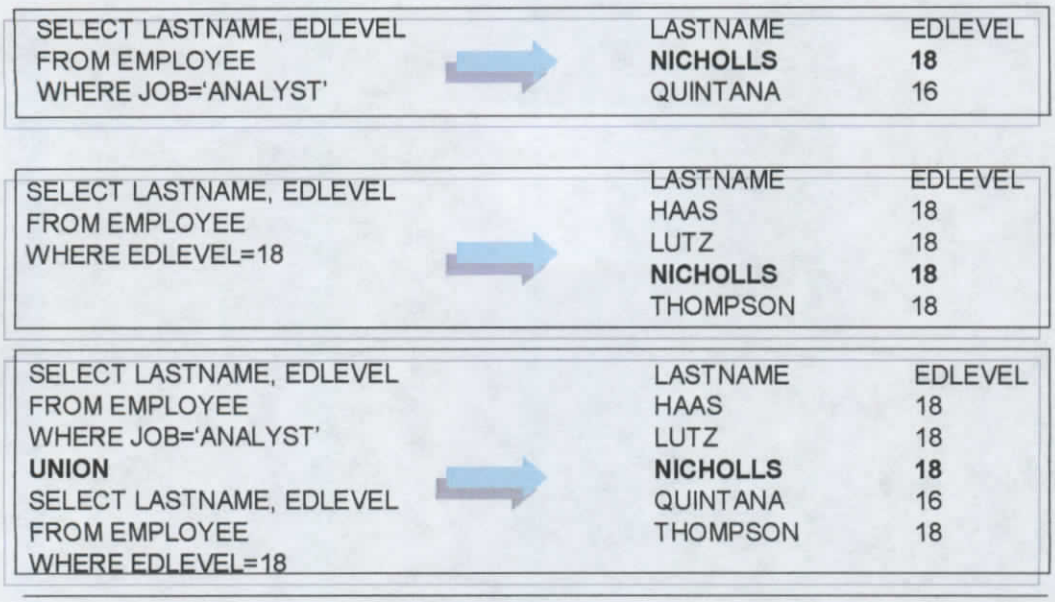


Figure 8-3 Combination of Multiple Result Tables

Notes:

UNION combines two sets of rows and removes duplicates.

The problem to list last name and education level of employees who are analysts or have an education level of 18 can be solved by means of a UNION performing the following three steps:

1. Each SELECT is evaluated separately
2. The results are merged and ordered
3. Duplicates are removed

UNION Details

- Each SELECT must have the same number of column
- Corresponding columns must have compatible data type
- UNION removes duplicates
- UNION performs an ascending sort including all columns in specified order
- If specified, ORDER BY must be the last cause of the statement

Figure 8-4 UNION Details

Notes:

UNION - Example



Each entry should contain two lines. The first line should include the manager's employee number and name; the second line the name of the department

```
SELECT MGRNO ,'Dept' , DEPTNAME
FROM DEPARTMENT
UNION
SELECT MGRNO, 'Mgrno' ,LASTNAME
FROM DEPARTMENT D, EMPLOYEE E
WHERE D.MGRNO=E.EMPNO
ORDER BY 1,2 DESC
```

MGRNO	DEPTNAME
000010	Mgr HAAS
000010	Dept SPIFFY COMPUTER SERVICE DIV.
000020	Mgr THOMPSON
000020	Dept PLANNING
000030	Mgr KWAN
000030	Dept INFORMATION CENTER
000050	Mgr GEYER
000050	Dept SUPPORT SERVICES

Figure 8-5 UNION - Example

Notes:

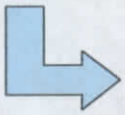
The first SELECT established the header of the result which is not really what the header should be.

Corresponding columns in the result tables to be combined by UNION must have compatible data types.

If a column name is the same for each SELECT, you can use the column name in the ORDER BY clause. Otherwise, you must specify the position number of the column in the result.

UNION – Generation of Fitting Result Rows

```
SELECT EMPNO,SUBSTR(FIRSTNME,1,1)||'.'|| MIDINIT,
        LASTNAME,SALARY AS INCME, 1 AS SORT
FROM EMPLOYEE
UNION
SELECT EMPNO,'',
        '',BONUS,2 AS SORT
FROM EMPLOYEE
UNION
SELECT EMPNO,'',
        '',COMM,3 AS SORT
FROM EMPLOYEE
UNION
SELECT EMPNO,'',
        'SUM:',SALARY+BONUS+COMM,
        4 AS SORT
FROM EMPLOYEE
ORDER BY EMPNO,SORT
```



EMPNO		LASTNAME	INCOME	SORT
000010	C.I	HAAS	52750.00	1
000010			1000.00	2
000010			4220.00	3
000010		SUM:	57970.00	4
000020	M.L	THOMPSON	41250.00	1
000020			800.00	2
000020			3300.00	3
000020		SUM:	45350.00	4

Figure 8-6 UNION – Generation of Fitting Result Rows

Notes:

Any type of SELECT statement can be used as long as the rules for UNION are adhered to. All column and scalar functions can be used.

In the above example, a substring of first name concatenated with the midint and the last name are selected in the first SELECT. To get the equivalent number of columns in the other SELECT statements, blanks (compatible to the datatypes of the concatenated value and the last name value) are specified as corresponding column values.

EMPNO must be selected to order the result table.

To get a four row result for each employee in the desired order
(1st salary, 2nd bonus, 3rd commission, 4th sum), we need an additional sort column.

The double exclamation points (!!), used in the above example, or double bars (||) are equivalent to the CONCAT operator.

UNION ALL- Principle

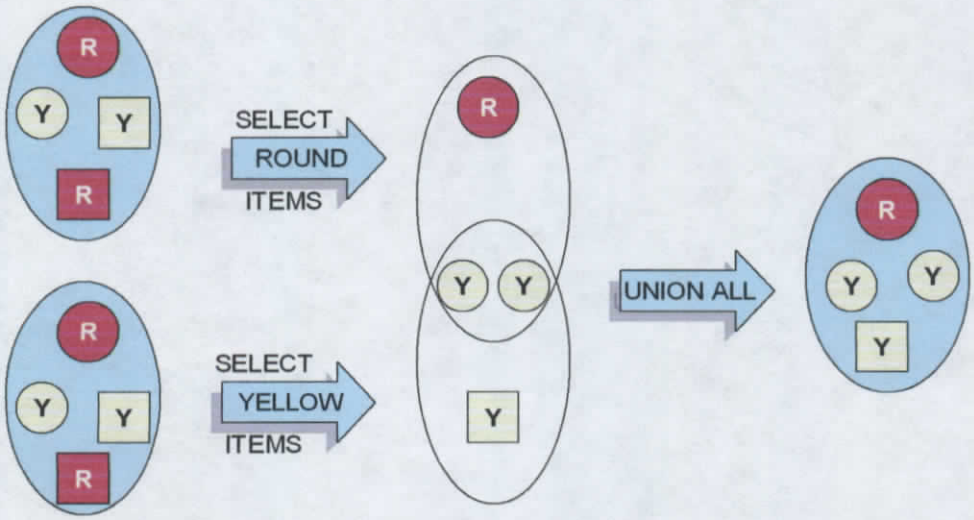


Figure 8-7 UNION ALL - Principle

Notes:

Each oval means a set of items (red and yellow squares and circles).

In contrast to mathematical set theory, UNION ALL combines two sets in the above example, i.e., it does not remove duplicates.

UNION ALL – Examples

```
SELECT LASTNAME, EDLEVEL
FROM EMPLOYEE
WHERE JOB='ANALYST'
```

LASTNAME	EDLEVEL
NICHOLLS	**18**
QUINTANA	16

```
SELECT LASTNAME, EDLEVEL
FROM EMPLOYEE
WHERE EDLEVEL=18
```

LASTNAME	EDLEVEL
HAAS	18
LUTZ	18
NICHOLLS	**18**
THOMPSON	18

```
SELECT LASTNAME, EDLEVEL
FROM EMPLOYEE
WHERE JOB='ANALYST'
UNION ALL
SELECT LASTNAME, EDLEVEL
FROM EMPLOYEE
WHERE EDLEVEL=18
```

LASTNAME	EDLEVEL
NICHOLLS	**18**
QUINTANA	16
HAAS	18
THOMPSON	18
NICHOLLS	**18**
LUTZ	18

Figure 8-8 UNION ALL – Examples

Notes:

UNION ALL does not eliminate duplicates.

UNION ALL often performs better than UNION.

Use UNION ALL if you are sure that there are no duplicates.

Summary

SELECTs with same number and compatible data type may be combined by use of keyword UNION

- UNION performs a sort to remove duplicates
- UNION ALL most performs better than UNION ALL
- Use UNION ALL, If not duplicates are expected
- Column numbers in the ORDER BY clause must be used when the column names in the SELECTs are different
- Each select may reference different tables or views

Figure 8-9 Summary

Notes:

Checkpoint

Exercise — Unit 8 UNION Checkpoint

1. The results of arbitrary SELECTs can be combined by means of UNION and ordered to get a new result.

   a. T

   b. F

2. Why or when should you use UNION?

   a. When all duplicate rows have to be eliminated.

   b. When SELECTs with an arbitrary number of columns should be combined.

3. Which rules do you have to consider when using UNION?

Unit 9 Using Subqueries

What This Unit Is About

This unit provides information on how to use subqueries.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Use subqueries in WHERE and HAVING conditions.
- Code subqueries using basic predicates.
- Code subqueries using the IN keyword.
- Code subqueries using the ALL and ANY/SOME keywords.

How You Will Check Your Progress

- Checkpoint questions
- Machine labs

Objectives

After completing this unit, you should be able to :

- Know, when to use subqueries
- Use subqueries in WHERE and HAVING clause
- Code subqueries using basic predicates
- Code subqueries using IN
- Code subqueries using ALL,ANY or SOME

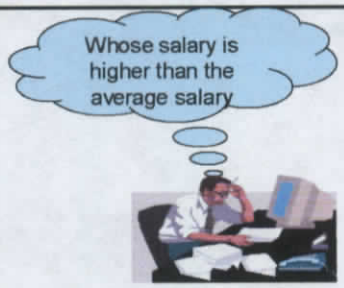
Figure 9-1 Objectives

Notes:

It is often useful to imbed a query within another. This enables the inner query to provide an answer to a question and use the result in the outer query to control the rows to be returned. The inner query is referred to as subquery.

9.1 Using Subqueries

Result Using Separate Selects




```
SELECT AVG(SALARY)
FROM EMPLOYEE
```

27303

```
SELECT
EMPNO,LASTNAME
FROM EMPLOYEE
WHERE SALARY >27303
```

Figure 9-2 Result Using Separate Selects

Notes:

Assume that you want to identify all employees whose salary is higher than the average salary of all employees.

Without subqueries, you would have to do the following:

- Evaluate the average salary of all employees (first select)
- Code a second query using the result in the WHERE condition.
- Run the second query to find the employees whose salary is higher than the average salary (27303).

Same Result Using a Subquery

```
SELECT EMPNO,LASTNAME
FROM EMPLOYEE
WHERE SALARY >(SELECT AVG(SALARY)
               FROM EMPLOYEE)
```

Figure 9-3 Same Result Using a Subquery

Notes:

The problem, described on the previous visual can be solved by means of a single query.

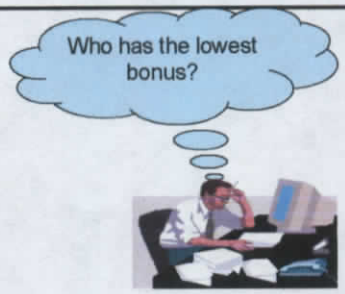
We can code the first SELECT of the previous visual in the WHERE condition of the second SELECT of the previous visual at the place where the average is needed.

As a consequence, the second SELECT is executed by the outer(main) query. The inner SELECT is referred to as the **subquery.**

The subquery runs first, and the result is used to evaluate the WHERE condition of the main query.

A subquery cannot contain the keywords UNION and ORDER BY.

Subquery with Basic Predicates



EMPNO	LASTNAME	BONUS
000010	HAAS	1000
000070	PULASKI	700
000290	PARKER	300

```
SELECT EMPNO,LASTNAME,BONUS
FROM EMPLOYEE
WHERE BONUS=(SELECT MIN(BONUS)
             FROM EMPLOYEE)
```

000290	PARKER	300

Figure 9-4 Subquery with Basic Predicates

Notes:

In this example, we look for the employees who have the lowest bonus.

Therefore, we must determine the lowest bonus and compare it with the bonuses of all employees. If the bonus of an employee matches the value provided by the subquery, the result of the predicate is true and the row is returned.

A basic predicate is one that uses =, <, >, or any combination thereof. A subquery in a basic predicate must not return more than one value, not have more than one row and more than one column.

Subquery – Set of Values

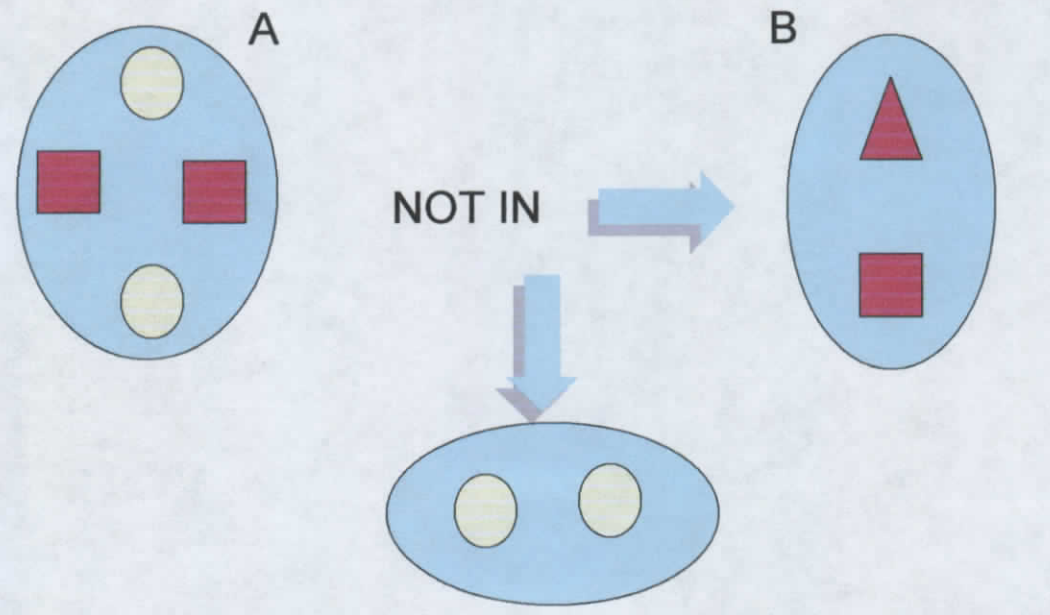


Figure 9-5 Subquery – Set of Values

Notes:

Assume that you have to identify all items of set A which are not in set B.

First, you execute a SELECT determining all items of set B (ITEMX, ITEMY,…)

Secondly, you use the list of items determined by the first query in the WHERE clause of a further SELECT:

WHERE ITEM NOT IN (ITEMX, ITEMY …)

Using subqueries, you can combine these two steps. The subquery will be executed first to retrieve the items of set B.

Subquery Using IN

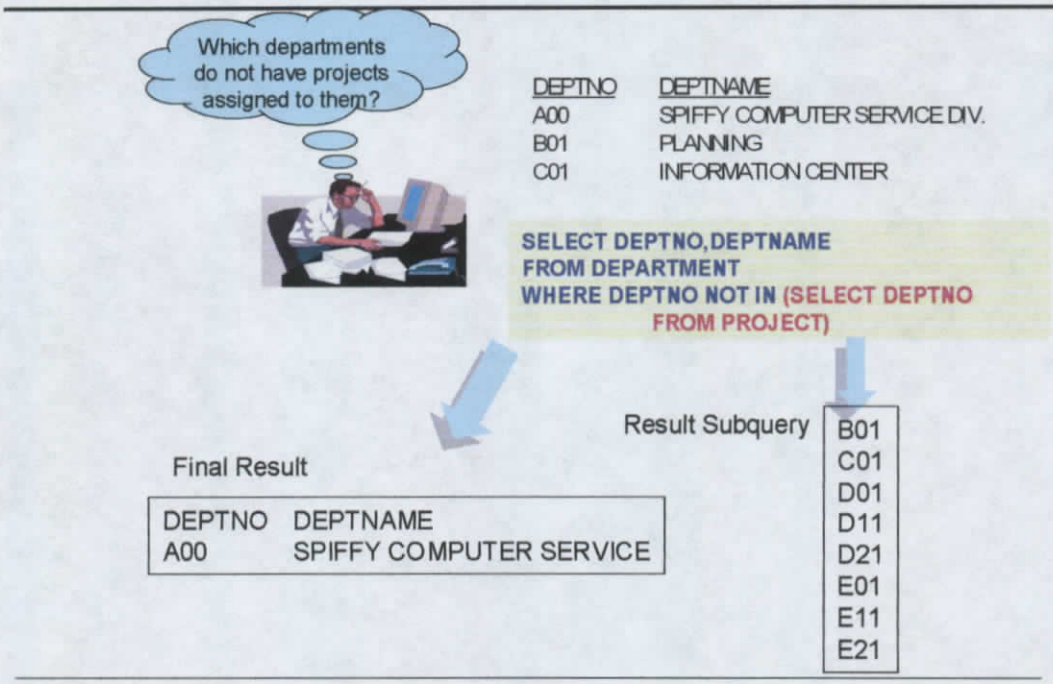


Figure 9-6 Subquery Using IN

Notes:

The subquery in the example builds a list of the DEPTNO values for every department of the PROJECT table. The outer query determines the department number and department name of any department whose department number is NOT in the list returned by the subquery. In the other words, the query is reporting on departments which do not have projects.

The IN predicate, when used with a subquery, enables the outer query to compare the values in a column of the outer table with a list of values provided by the subquery.

A subquery used with an IN predicate must identify a single result column and may return any number of values.

For a IN predicate, if any of the non-null values returned by the subquery match any of the non-null values of the column being searched in the outer query, the rows in the outer query that contain the matching value will appear in the final result set.
Nulls in the subquery result will never match nulls in the outer query result.

When the IN is prefaced with a NOT, as in the example, the subquery should be written to return only non-null values. Then, all of the rows of the outer query that do not match any of the values returned by the subquery will appear in the result set.
If the NOT IN subquery return a null, the outer query will always produce an empty result set.

Subquery Using ALL

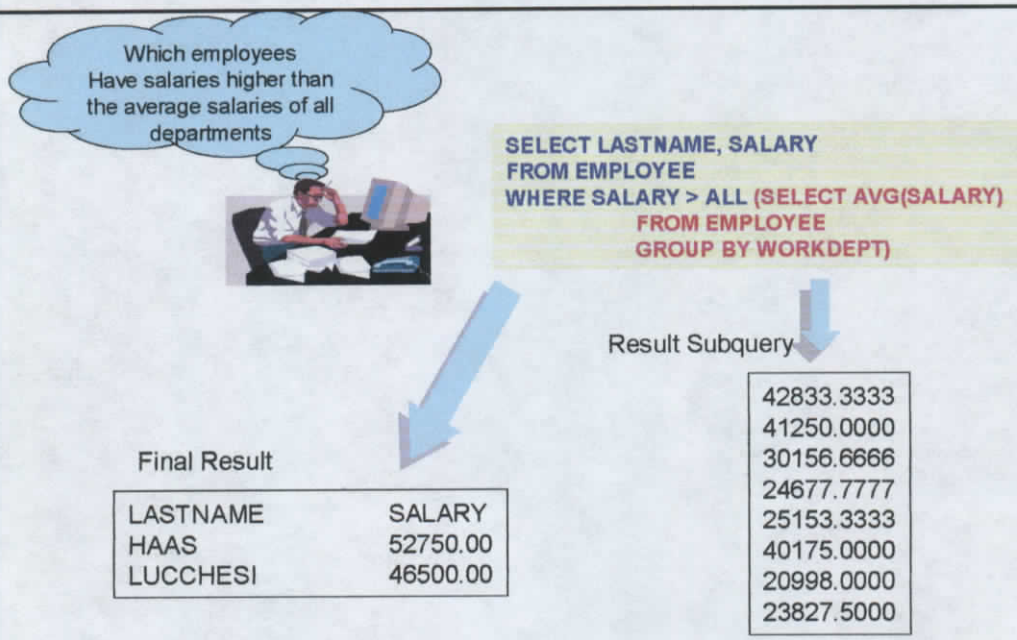


Figure 9-7 Subquery Using ALL

Notes:

A quantified predicates compares each values in the outer query with the set of values provided by the subquery. A quantified predicates is one that uses one of the following keywords: ALL, ANY, or SOME. As usual, the subquery must specify a single result column and may return any number of values.

The query in the example runs in two steps, same as in prior examples.

1. The subquery runs to find the average salary for each department.
2. Each employee's salary is compared with the list of department average. All employees are listed whose salary is higher than all the values in the set provided by the subquery

Subquery Using ANY or SOME

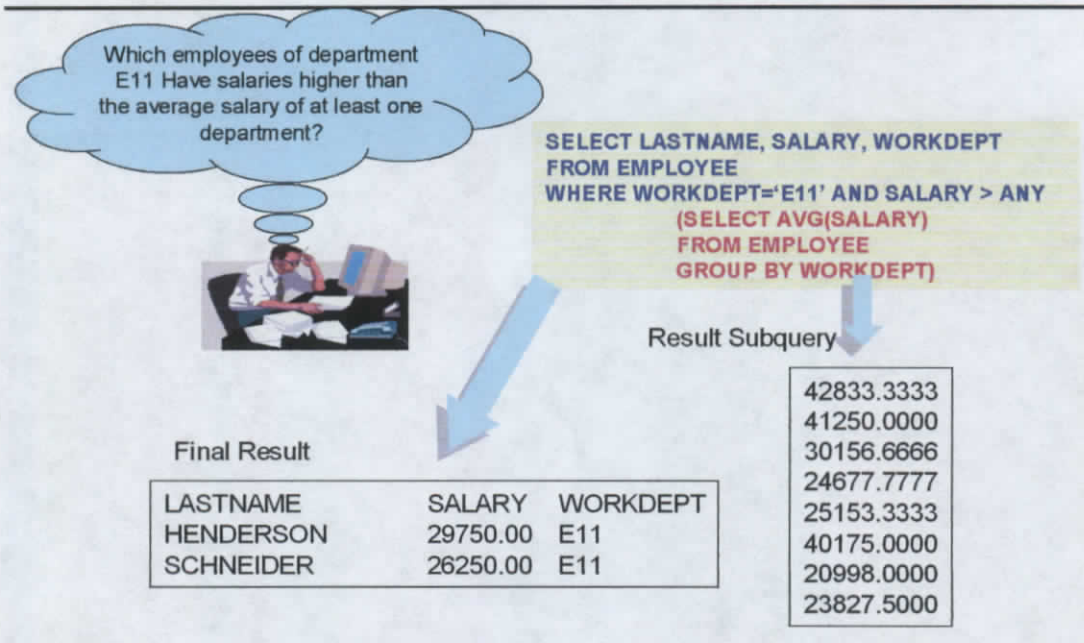


Figure 9-8 Subquery Using ANY or SOME

Notes:

When the **ANY** or **SOME** keywords are specified - they have exact same meaning and effect.

The query in the example runs in two steps, same as in the ALL example.

1. The subquery runs to find the average salary for each department
2. Each employee's salary is compared with the list of department average. All employees are listed whose salary is higher than at least one of the values in the set provided by the subquery

Subquery within HAVING Clause

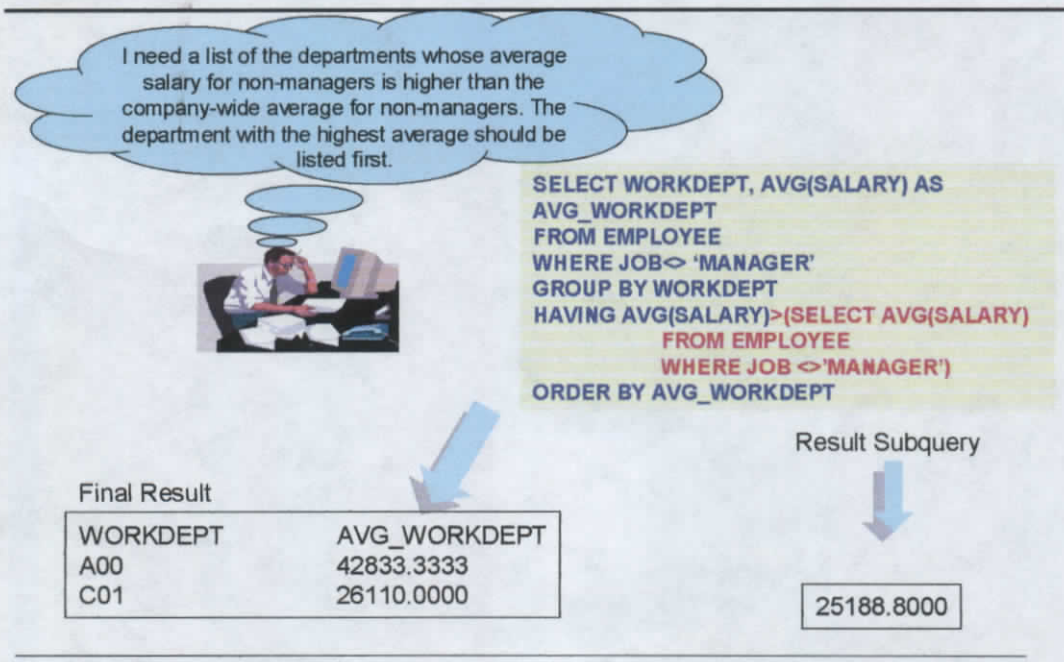


Figure 9-9 Subquery within HAVING Clause

Notes:

Subqueries can be also used in the HAVING condition for GROUP values.

Summary

Subqueries
- Must be on right side of search condition when used in WHERE clause or in HAVING clause
- Must be enclosed in parentheses
- Can return single or multiple values from only one column
- Cannot contain UNION, UNION ALL, or ORDER BY
- Number of values subquery can return determines operator in outer SELECT.

Figure 9-10 Summary

Notes:

Checkpoint

Exercise — Unit 9 Using Subqueries Checkpoint

1. Subqueries can be used in WHERE or HAVING clauses of a SELECT.

   a. T

   b. F

2. If you want to list the employees whose salary is the highest, you may code in the WHERE condition.

   a. SALARY > (SELECT SALARY FROM EMPLOYEE...)

   b. SALARY = (SELECT MAX(SALARY) FROM EMPLOYEE...)

   c. SALARY > ALL(SELECT SALARY FROM EMPLOYEE...)

3. List three keywords which can quantify predicates when using Subqueries.

   _____

   _____

   _____

Unit 10 Maintaining Data

What This Unit Is About

In addition to the SELECT statement, the SQL Data Manipulation Language (DML) consists of three statements. These are used to add rows to a table (INSERT), modify data within existing rows (UPDATE), and remove rows (DELETE).

What You Should Be Able to Do

After completing this unit, you should be able to:

- Create a simple table for testing DML statements
- Insert rows into a table.
- Update data of a table
- Delete rows from a table

How You Will Check Your Progress

- Checkpoint questions
- Machine labs

Objectives

After completing this unit, you should be able to :

- Create a simple table to test DML statement
- Create a view
- Insert rows into a table
- Update data in a table
- Delete rows from a table
- Drop a table

Figure 10-1 Objectives

Notes:

Creating tales is normally done by database administrators. For test purpose, it could be helpful to create a simple test table by yourself.

After the test table has being created, INSERT, UPDATE and DELETE statement can be used to insert, modify, and delete rows or values

10.1 Maintaining Data

TESTEMP Table

EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
10	HAAS	A00	01/01/1965	52750	1000
20	THOMPSON	B01	10/10/1973	41250	800
30	KWAN	C01	05/04/1975	38250	800
50	GEYER	E01	17/08/1949	40175	800
111	SMITH	C01	25/06/1998	25000	

Figure 10-2 TESTEMP Table

Notes:

The visual shows the contest of the TESTEMP table. This table will be the basis for the exercises of this unit.

Create table TESTEMP

```
CREATE TABLE EMPLOYEE

(EMPNO        CHAR(6)        NOT NULL,
LASTNAME      VARCHAR(15)    NOT NULL,
WORKDEPT      CHAR(3)        ,
HIREDATE      DATE           ,
SALARY        DECIMAL(9,2)   ,
BONUS         DECIMAL(9,2)   )
```

Figure 10-3 Create table TESTEMP

Notes:

This CREATE TABLE statement creates the TESTEMP table.

In addition to the column definition, an IN clause can be required. The IN clause depends on the environment.

Create View PERSINFO

```
CREATE VIEW PERSINFO AS
SELECT  EMPNO,LASTNAME,
        SALARY AS YEAR_SAL
FROM    TESTEMP
WHERE   WORKDEPT ='C01'

SELECT *
FROM PERSINFO
```



EMPNO	LASTNAME	YEAR_SAL
000030	KWAN	38250.00
000111	SMITH	25000.00

Figure10-4 Create View PERSINFO

Notes:

This CREATE VIEW statement describes a part of the base table, consisting of employee number, last name, and yearly salary of all employees in department C01.
The AS clause in the subselect enables you to specify column names in the view different from the column names in the referenced table. In the above example, "SALARY AS YEAR_SAL" is used to rename the SALARY column in the base table TESTEMP to YEAR_SAL in view PERSINFO. Users referencing the view PERSINFO must use the new column name.
Another possibility for renaming columns is in a view is:

```
CREATE VIEW PERSINFO (EMPNO, LASTNAME, YEAR_SAL)
        AS SELECT EMPNO, LASTNAME, SALARY
        FROM EMPLOYEE WHERE WORKDEPT='C01'
```

When using this syntax, the number of column names listed in the parentheses must match the number of the selected columns.
Users who are authorized to use view PERSINFO will get the listed columns for employees whose department number is C01. Other data is not visible.

Insert a Single Row

```
INSERT INTO TESTEMP
VALUES ('000111','SMITH','C01','1998-06-25',25000,NULL)

OR

INSERT INTO TESTEMP
(EMPNO,LASTNAME,WORKDEPT,HIREDATE,SALARY)
VALUES ('000111','SMITH','C01','1998-06-25',25000)
```



EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
000111	SMITH	C01	1998-06-25	25000.00	----

Figure 10-5 Insert a Single Row

Notes:

Use an INSERT statement to insert rows into a table. Inserting a row a view inserts the row into the table on which he view is based. Therefore, inserts via a view are only possible if the view contains all columns that are defined with NOT NULL in the base table.

In the first example, the column list is omitted. That means an implicit specification of the column list. In this case, the column list contains all columns of the base table, in the order they are defined for the base table. Therefore, the VALUES clause must provide values for all columns in that order.

In the second example, a column list is specified. The number of values provided for a row via the VALUES clause must be equal to the number of column names in the column list. The first value is inserted in the first column of the list, the second value in the second column, and so on.

Insert Multiple Row

TESTEMP
EMPNO,LASTNAME,WORKDEPT,HIREDATE,SALARY,BONUS

```
INSERT INTO TESTEMP
SELECT EMPNO,LASTNAME,WORKDEPT,HIREDATE,SALARY,BONUS
FROM EMPLOYEE
WHERE EMPNO<='000050'
```



EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
10	HAAS	A00	01/01/1965	52750	1000
20	THOMPSON	B01	10/10/1973	41250	800
30	KWAN	C01	05/04/1975	38250	800
50	GEYER	E01	17/08/1949	40175	800
111	SMITH	C01	25/06/1998	25000	

Figure 10-6 Insert Multiple Row

Notes:

Instead of the VALUES clause, include a SELECT statement in the INSERT statement to tell DB2 that another table or view contains the data for the new rows.

In the example, the number and sequence of the selected columns of the EMPLOYEE table must match the number and sequence of the columns of the TESTEMP table.
If a columns list for the target table is used, all its NOT NULL columns must be included.

Update Columns

EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
10	HAAS	A00	01/01/1965	52750	1000
20	THOMPSON	B01	10/10/1973	41250	800
30	KWAN	C01	05/04/1975	38250	800
50	GEYER	E01	17/08/1949	40175	800
111	SMITH	C01	25/06/1998	25000	

```
UPDATE TESTEMP
SET BONUS =500
SALARY=26000
WHERE EMPNO ='000111'
```



EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
10	HAAS	A00	01/01/1965	52750	1000
20	THOMPSON	B01	10/10/1973	41250	800
30	KWAN	C01	05/04/1975	38250	800
50	GEYER	E01	17/08/1949	40175	800
111	SMITH	C01	25/06/1998	26000	500

Figure 10-7 Update Columns

Notes:

The UPDATE statement updates the values of the specified columns for one ore more rows of a table or view.

Our new colleague, Mr. Smith (employee number 000111) receives a first bonus and a salary raise. Therefore, we must set BONUS to 500 and SALARY to 26000.

If more than one column is changed, the columns must be separated by commas. The SET keyword must precede the first column and must not be repeated.

To set BONUS to NULL, the UPDATE statement must be coded as follows.

```
UPDATE TESTEMP
SET BONUS = NULL
WHERE EMPNO= '000111'
```

Update Columns (Cont'd)

EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
10	HAAS	A00	01/01/1965	52750	1000
20	THOMPSON	B01	10/10/1973	41250	800
30	KWAN	C01	05/04/1975	38250	800
50	GEYER	E01	17/08/1949	40175	800
111	SMITH	C01	25/06/1998	25000	

```
UPDATE TESTEMP
SET SALARY=SALARY +1000
WHERE WORKDEPT ='C01'
```



EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
10	HAAS	A00	01/01/1965	52750	1000
20	THOMPSON	B01	10/10/1973	41250	800
30	KWAN	C01	05/04/1975	39250	800
50	GEYER	E01	17/08/1949	40175	800
111	SMITH	C01	25/06/1998	26000	

Figure 10-8 Update Columns (Cont'd)

Notes:

Our department C01 has finished a project with great success. The salary of all employees should be increased by $1,000.

To update selected rows, use the WHERE clause to specify the rows to be updated.

If you **omit the WHERE clause, all rows** of the table or view are updated.

Delete Rows

EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
10	HAAS	A00	01/01/1965	52750	1000
20	THOMPSON	B01	10/10/1973	41250	800
30	KWAN	C01	05/04/1975	38250	800
50	GEYER	E01	17/08/1949	40175	800
111	SMITH	C01	25/06/1998	25000	

```
DELETE FROM TESTEMP
WHERE EMPNO ='000111'
```



EMPNO	LASTNAME	WORKDEPT	HIREDATE	SALARY	BONUS
10	HAAS	A00	01/01/1965	52750	1000
20	THOMPSON	B01	10/10/1973	41250	800
30	KWAN	C01	05/04/1975	38250	800
50	GEYER	E01	17/08/1949	40175	800

Figure 10-9 Delete Rows

Notes:

Mr. Smith left the company. Therefore we delete the row with employee number 000111.

The DELETE statement removes entire rows from a table. Depending on the search condition of the WHERE clause, the DELETE statement removes zero, one, of more rows of a table.

To delete specific rows, use the WHERE clause to specify the rows to be deleted.

If you **omit the WHERE clause, all** rows of the table or view are **deleted.**

Drop Table TESTEMP

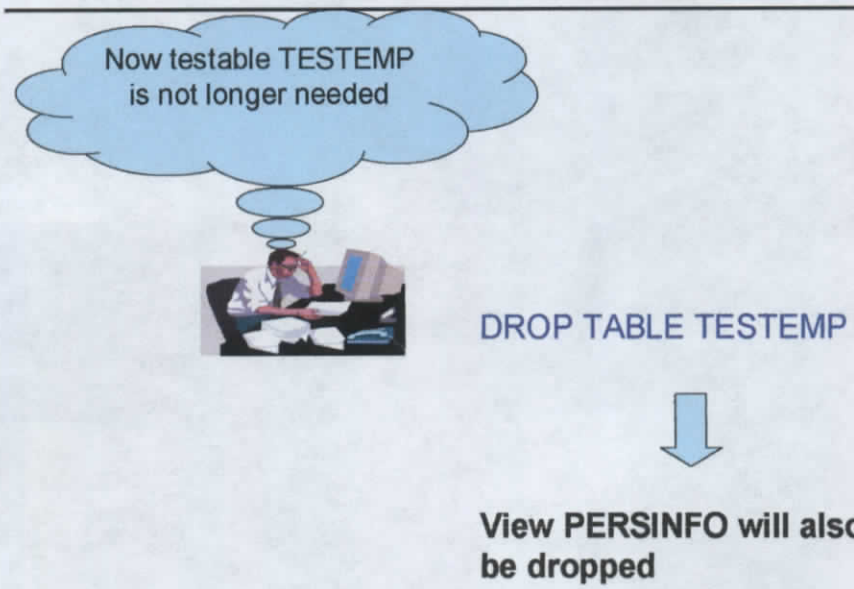


Figure 10-10 Drop Table TESTEMP

Notes:

When executing the statement DROP TESTEMP, DB2 deletes the object, i.e., the table.
As a consequence, all rows and the table definition are deleted.

Any objects (e.g., view, indexes) that are dependent on the table are deleted as well.

Summary

- You must be authorized, to execute CREATE/DROP table statement
- You must authorized, to perform statements which change the content of a table
- Dropping a table, drop objects depending on that table too
- Search condition in the WHERE clause specifies the rows to be updated or deleted
- Inserting a rows into a view also inserts a row into the table on which the vies is based
- Updating a row of a view updates a row of the table on which the view is based
- Omitting the WHERE clause in an UPDATE or DELETE statement cause all rows of the table or view are updated or deleted

Figure 10-11 Summary

Notes:

Checkpoint

Exercise — Unit 10 Maintaining Data Checkpoint

1. You must explicitly be authorized to INSERT data rows in your own table.

   a. T

   b. F

2. To delete all rows of a table its description, you must execute.

   a. DROP TABLE table name

   b. DELETE FROM table name and DROP TABLE table name

   c. DELETE FROM table name

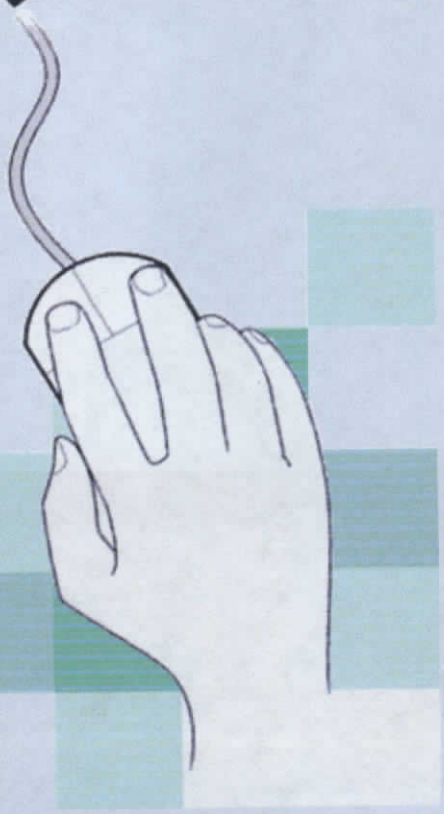
PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

SEDE - AMBATO





Material Educativo





DB2 SQL Workshop

Student Exercises

Material de Certificación

Contents

Table Relationship iv

Table Descriptions v

Table Content vii

EXERCISES

Exercise 4. Simple SQL Queries 4-1

Exercise 5. Retrieving Data from Multiple Tables 5-1

Exercise 6. Scalar Functions and Arithmetic 6-1

Exercise 7. Column Functions and Grouping 7-1

Exercise 8. UNION 8-1

Exercise 9.Using Subqueries 9-1

Exercise 10. Maintaining Data 10-1

Table Relationships

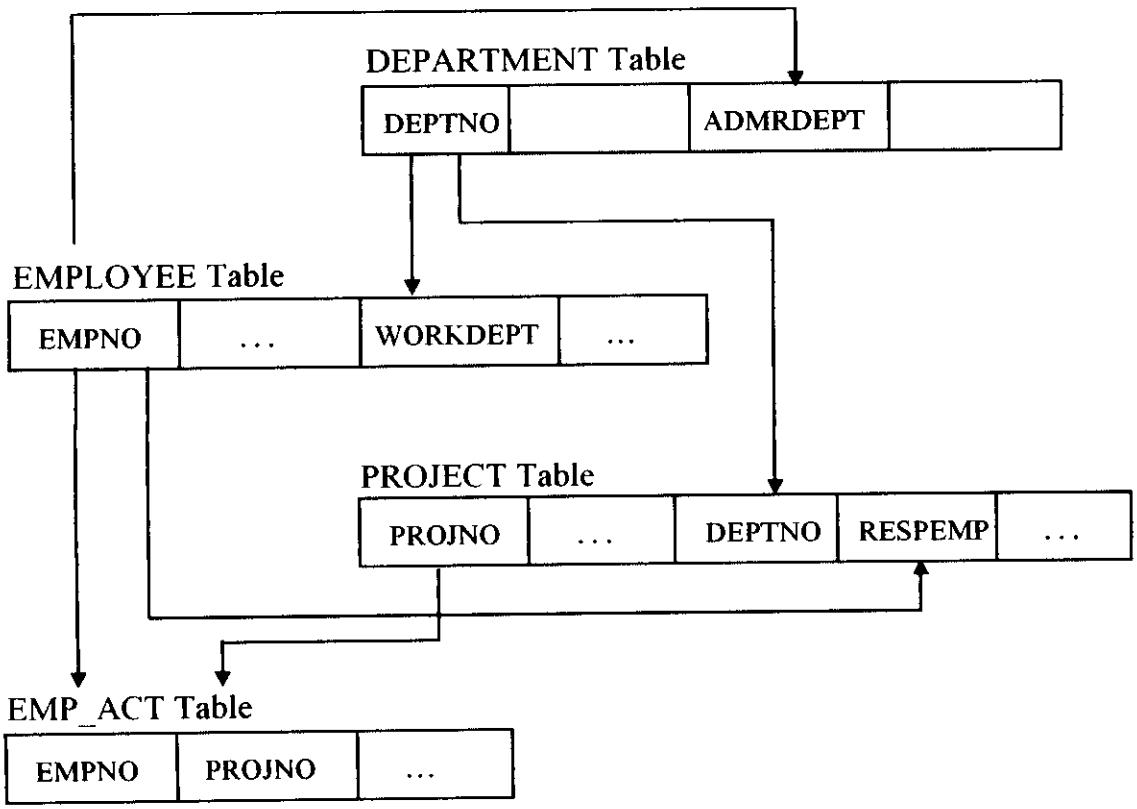


Figure 0-1. Table Relationships

Notes:

This diagram illustrates the relationships between the tables used in the exercises for this course. The lines show the connection between the tables.

Table Descriptions

The tables are described in hierarchical order, as shown in the Table Relationships on the previous page

DEPARTMENT

There is one row in the DEPARTMENT table for each department in the company.

Column Name	Meaning	Data Type	NULLS allowed
DEPTNO	Department Number	CHAR(3)	N
DEPTNAME	Department Name	VARCHAR(36)	N
MGRNO	Employee Number of the Responsible Manager	CHAR(6)	Y
ADMRDEPT	Department Number of the Department to which the Department reports	CHAR(3)	N
LOCATION	Location Number	CHAR(5)	Y

EMPLOYEE

There is one row in the EMPLOYEE table for each of the employees in the company.

Column Name	Meaning	Data Type	NULLS allowed
EMPNO	Employee Number	CHAR(6)	N
FIRSTNME	First Name	VARCHAR(20)	N
MIDINIT	Middle Initial	CHAR(1)	N
LASTNAME	Last Name	VARCHAR(15)	N
WORKDEPT	Department in which the employee works	CHAR(3)	Y
PHONENO	Phone Number	CHAR(4)	Y
HIREDATE	Date of the Hire	DATE	Y
JOB	Job	CHAR(8)	Y
EDLEVEL	Number of Years of Formal Education	SMALLINT	Y
SEX	Sex (M male, F female)	CHAR(1)	Y
BIRTHDATE	Date of Birth	DATE	Y
SALARY	Yearly Salary	DECIMAL(9,2)	Y
BONUS	Yearly Bonus	DECIMAL(9,2)	Y
COMM	Yearly Commission	DECIMAL(9,2)	Y

PROJECT

There is one row in the PROJECT table for each project.

Column Name	Meaning	Data Type	NULLS allowed
PROJNO	Project Number	CHAR(6)	N
PROJNAME	Project Name	VARCHAR(24)	N
DEPTNO	Responsible Department	CHAR(3)	N
RESPEMP	Employee Number of the Responsible Employee	CHAR(6)	N
PRSTAFF	Estimated Mean Staffing	DECIMAL(5,2)	Y
PRSTDATE	Estimated Start Date	DATE	Y
PRENDATE	Estimated End Date	DATE	Y
MAJPROJ	Major Project for a Subproject	CHAR(6)	Y

EMP_ACT

There are many rows in the EMP_ACT table for any employee or any project.

Column Name	Meaning	Data Type	NULLS allowed
EMPNO	Employee Number of Employee Performing the Activity	CHAR(6)	N
PROJNO	Project Number	CHAR(6)	N
ACTNO	Activity Number	SMALLINT	N
EMPTIME	Proportion of Employee's Time Spent on Project	DECIMAL(5,2)	Y
EMSTDATE	Date Activity Starts	DATE	Y
EMENDATE	Date Activity Ends	DATE	Y

Table Content

DEPARTMENT table

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	-
B01	PLANNING	000020	A00	-
C01	INFORMATION CENTER	000030	A00	-
D01	DEVELOPMENT CENTER	-	A00	-
D11	MANUFACTURING SYSTEMS	000060	D01	-
D21	ADMINISTRATION SYSTEMS	000070	D01	-
E01	SUPPORT SERVICES	000050	A00	-
E11	OPERATIONS	000090	E01	-
E21	SOFTWARE SUPPORT	000100	E01	-

9 registro(s) seleccionados.

EMPLOYEE table

EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
000010	CHRISTINE	I	HAAS	A00	3978	1/01/1965
000020	MICHAEL	L	THOMPSON	B01	3476	0/10/1973
000030	SALLY	A	KWAN	C01	4738	5/04/1975
000050	JOHN	B	GEYER	E01	6789	7/08/1949
000060	IRVING	F	STERN	D11	6423	4/09/1973
000070	EVA	D	PULASKI	D21	7831	0/09/1980
000090	EILEEN	W	HENDERSON	E11	5498	5/08/1970
000100	THEODORE	Q	SPENSER	E21	0972	9/06/1980
000110	VINCENZO	G	LUCCHESSI	A00	3490	6/05/1958
000120	SEAN		O'CONNELL	A00	2167	5/12/1963
000130	DOLORES	M	QUINTANA	C01	4578	8/07/1971
000140	HEATHER	A	NICHOLLS	C01	1793	5/12/1976
000150	BRUCE		ADAMSON	D11	4510	2/02/1972
000160	ELIZABETH	R	PIANKA	D11	3782	1/10/1977
000170	MASATOSHI	J	YOSHIMURA	D11	2890	5/09/1978
000180	MARILYN	S	SCOUTTEN	D11	1682	7/07/1973
000190	JAMES	H	WALKER	D11	2986	6/07/1974
000200	DAVID		BROWN	D11	4501	3/03/1966
000210	WILLIAM	T	JONES	D11	0942	1/04/1979
000220	JENNIFER	K	LUTZ	D11	0672	9/08/1968
000230	JAMES	J	JEFFERSON	D21	2094	1/11/1966
000240	SALVATORE	M	MARINO	D21	3780	5/12/1979
000250	DANIEL	S	SMITH	D21	0961	0/10/1969
000260	SYBIL	P	JOHNSON	D21	8953	1/09/1975
000270	MARIA	L	PEREZ	D21	9001	0/09/1980
000280	ETHEL	R	SCHNEIDER	E11	8997	4/03/1967
000290	JOHN	R	PARKER	E11	4502	0/05/1980
000300	PHILIP	X	SMITH	E11	2095	9/06/1972
000310	MAUDE	F	SETRIGHT	E11	3332	2/09/1964
000320	RAMLAL	V	MEHTA	E21	9990	7/07/1965
000330	WING		LEE	E21	2103	3/02/1976
000340	JASON	R	GOUNOT	E21	5698	5/05/1947

EMPLOYEE table (cont'd)

JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
PRES	18	F	24/08/1933	52750,00	1000,00	4220,00
MANAGER	18	M	02/02/1948	41250,00	800,00	3300,00
MANAGER	20	F	11/05/1941	38250,00	800,00	3060,00
MANAGER	16	M	15/09/1925	40175,00	800,00	3214,00
MANAGER	16	M	07/07/1945	32250,00	500,00	2580,00
MANAGER	16	F	26/05/1953	36170,00	700,00	2893,00
MANAGER	16	F	15/05/1941	29750,00	600,00	2380,00
MANAGER	14	M	18/12/1956	26150,00	500,00	2092,00
SALESREP	19	M	05/11/1929	46500,00	900,00	3720,00
CLERK	14	M	18/10/1942	29250,00	600,00	2340,00
ANALYST	16	F	15/09/1925	23800,00	500,00	1904,00
ANALYST	18	F	19/01/1946	28420,00	600,00	2274,00
DESIGNER	16	M	17/05/1947	25280,00	500,00	2022,00
DESIGNER	17	F	12/04/1955	22250,00	400,00	1780,00
DESIGNER	16	M	05/01/1951	24680,00	500,00	1974,00
DESIGNER	17	F	21/02/1949	21340,00	500,00	1707,00
DESIGNER	16	M	25/06/1952	20450,00	400,00	1636,00
DESIGNER	16	M	29/05/1941	27740,00	600,00	2217,00
DESIGNER	17	M	23/02/1953	18270,00	400,00	1462,00
DESIGNER	18	F	19/03/1948	29840,00	600,00	2387,00
CLERK	14	M	30/05/1935	22180,00	400,00	1774,00
CLERK	17	M	31/03/1954	28760,00	600,00	2301,00
CLERK	15	M	12/11/1939	19180,00	400,00	1534,00
CLERK	16	F	05/10/1936	17250,00	300,00	1380,00
CLERK	15	F	26/05/1953	27380,00	500,00	2190,00
OPERATOR	17	F	28/03/1936	26250,00	500,00	2100,00
OPERATOR	12	M	09/07/1946	15340,00	300,00	1227,00
OPERATOR	14	M	27/10/1936	17750,00	400,00	1420,00
OPERATOR	12	F	21/04/1931	15900,00	300,00	1272,00
FIELDREP	16	M	11/08/1932	19950,00	400,00	1596,00
FIELDREP	14	M	18/07/1941	25370,00	500,00	2030,00
FIELDREP	16	M	17/05/1926	23840,00	500,00	1907,00

```
32 registro(s) seleccionados.
```

PROJECT table

```
PROJNO PROJNAME            DEPT        PR
                           NO  RESPEMP STAFF PRSTDATE PRENDATE  MAJPROJ
---------- ------ ------- ------- ---------- ---------- -------
AD3100 ADMIN SERVICES      D01 000010 6,50 01/01/1982 01/02/1983 -
AD3110 GENERAL ADMIN SYST  D21 000070 6,00 01/01/1982 01/02/1983 AD3100
AD3111 PAYROLL PROGRAMMIN  D21 000230 2,00 01/01/1982 01/02/1983 AD3110
AD3112 PERSONNEL PROGRAMM  D21 000250 1,00 01/01/1982 01/02/1983 AD3110
AD3113 ACCOUNT PROGRAMMIN  D21 000270 2,00 01/01/1982 01/02/1983 AD3110
IF1000 QUERY SERVICES      C01 000030 2,00 01/01/1982 01/02/1983 -
IF2000 USER EDUCATION      C01 000030 1,00 01/01/1982 01/02/1983 -
MA2100 WELD LINE AUTOMATI  D01 00001012,00 01/01/1982 01/02/1983 -
MA2110 W L PROGRAMMING     D11 000060 9,00 01/01/1982 01/02/1983 MA2100
MA2111 W L PROGRAM DESIG   D11 000220 2,00 01/01/1982 01/12/1982 MA2110
MA2112 W L ROBOT DESIGN    D11 000150 3,00 01/01/1982 01/12/1982 MA2110
MA2113 W L PROD CONT PR    D11 000160 3,00 15/02/1982 01/12/1982 MA2110
OP1000 OPERATION SUPPOR    E01 000050 6,00 01/01/1982 01/02/1983 -
OP1010 OPERATION           E11 000090 5,00 01/01/1982 01/02/1983 OP1000
OP2000 GEN SYSTEMS SERV    E01 000050 5,00 01/01/1982 01/02/1983 -
OP2010 SYSTEMS SUPPORT     E21 000100 4,00 01/01/1982 01/02/1983 OP2000
OP2011 SCP SYSTEMS SUPPO   E21 000320 1,00 01/01/1982 01/02/1983 OP2010
OP2012 APPLICATIONS SUPP   E21 000330 1,00 01/01/1982 01/02/1983 OP2010
OP2013 DB/DC SUPPORT       E21 000340 1,00 01/01/1982 01/02/1983 OP2010
PL2100 WELD LINE PLANNIN   B01 000020 1,00 01/01/1982 15/09/1982 MA2100


20 registro(s) seleccionados.
```

EMP_ACT table

EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
000010	MA2100	10	0,50	01/01/1982	01/11/1982
000010	MA2110	10	1,00	01/01/1982	01/02/1983
000010	AD3100	10	0,50	01/01/1982	01/07/1982
000020	PL2100	30	1,00	01/01/1982	15/09/1982
000030	IF1000	10	0,50	01/06/1982	01/01/1983
000030	IF2000	10	0,50	01/01/1982	01/01/1983
000050	OP1000	10	0,25	01/01/1982	01/02/1983
000050	OP2010	10	0,75	01/01/1982	01/02/1983
000070	AD3110	10	1,00	01/01/1982	01/02/1983
000090	OP1010	10	1,00	01/01/1982	01/02/1983
000100	OP2010	10	1,00	01/01/1982	01/02/1983
000110	MA2100	20	1,00	01/01/1982	01/03/1982
000130	IF1000	90	1,00	01/01/1982	01/10/1982
000130	IF1000	100	0,50	01/10/1982	01/01/1983
000140	IF1000	90	0,50	01/10/1982	01/01/1983
000140	IF2000	100	1,00	01/01/1982	01/03/1982
000140	IF2000	100	0,50	01/03/1982	01/07/1982
000140	IF2000	110	0,50	01/03/1982	01/07/1982
000140	IF2000	110	0,50	01/10/1982	01/01/1983
000150	MA2112	60	1,00	01/01/1982	15/07/1982
000150	MA2112	180	1,00	15/07/1982	01/02/1983
000160	MA2113	60	1,00	15/07/1982	01/02/1983
000170	MA2112	60	1,00	01/01/1982	01/06/1983
000170	MA2112	70	1,00	01/06/1982	01/02/1983
000170	MA2113	80	1,00	01/01/1982	01/02/1983
000180	MA2113	70	1,00	01/04/1982	15/06/1982
000190	MA2112	70	1,00	01/02/1982	01/10/1982
000190	MA2112	80	1,00	01/10/1982	01/10/1983
000200	MA2111	50	1,00	01/01/1982	15/06/1982
000200	MA2111	60	1,00	15/06/1982	01/02/1983
000210	MA2113	80	0,50	01/10/1982	01/02/1983
000210	MA2113	180	0,50	01/10/1982	01/02/1983
000220	MA2111	40	1,00	01/01/1982	01/02/1983
000230	AD3111	60	1,00	01/01/1982	15/03/1982
000230	AD3111	60	0,50	15/03/1982	15/04/1982
000230	AD3111	70	0,50	15/03/1982	15/10/1982
000230	AD3111	80	0,50	15/04/1982	15/10/1982
000230	AD3111	180	1,00	15/10/1982	01/01/1983
000240	AD3111	70	1,00	15/02/1982	15/09/1982
000240	AD3111	80	1,00	15/09/1982	01/01/1983
000250	AD3112	60	1,00	01/01/1982	01/02/1982
000250	AD3112	60	0,50	01/02/1982	15/03/1982
000250	AD3112	60	0,50	01/12/1982	01/01/1983
000250	AD3112	60	1,00	01/01/1983	01/02/1983
000250	AD3112	70	0,50	01/02/1982	15/03/1982

EMP_ACT table (cont'd)

```
000250 AD3112     70    1,00 15/03/1982 15/08/1982
000250 AD3112     70    0,25 15/08/1982 15/10/1982
000250 AD3112     80    0,25 15/08/1982 15/10/1982
000250 AD3112     80    0,50 15/10/1982 01/12/1982
000250 AD3112    180    0,50 15/08/1982 01/01/1983
000260 AD3113     70    0,50 15/06/1982 01/07/1982
000260 AD3113     70    1,00 01/07/1982 01/02/1983
000260 AD3113     80    1,00 01/01/1982 01/03/1982
000260 AD3113     80    0,50 01/03/1982 15/04/1982
000260 AD3113    180    0,50 01/03/1982 15/04/1982
000260 AD3113    180    1,00 15/04/1982 01/06/1982
000260 AD3113    180    0,50 01/06/1982 01/07/1982
000270 AD3113     60    0,50 01/03/1982 01/04/1982
000270 AD3113     60    1,00 01/04/1982 01/09/1982
000270 AD3113     60    0,25 01/09/1982 15/10/1982
000270 AD3113     70    0,75 01/09/1982 15/10/1982
000270 AD3113     70    1,00 15/10/1982 01/02/1983
000270 AD3113     80    1,00 01/01/1982 01/03/1982
000270 AD3113     80    0,50 01/03/1982 01/04/1982
000280 OP1010    130    1,00 01/01/1982 01/02/1983
000290 OP1010    130    1,00 01/01/1982 01/02/1983
000300 OP1010    130    1,00 01/01/1982 01/02/1983
000310 OP1010    130    1,00 01/01/1982 01/02/1983
000320 OP2011    140    0,75 01/01/1982 01/02/1983
000320 OP2011    150    0,25 01/01/1982 01/02/1983
000330 OP2012    140    0,25 01/01/1982 01/02/1983
000330 OP2012    160    0,75 01/01/1982 01/02/1983
000340 OP2013    140    0,50 01/01/1982 01/02/1983
000340 OP2013    170    0,50 01/01/1982 01/02/1983
000020 PL2100     30    1,00 01/01/1982 15/09/1982

  75 registro(s) seleccionados.
```

Exercise 4. Simple SQL Queries

What This Exercise is About

This exercise provides an opportunity to perform some basic SQL statements.

What You Should Be Able to Do

At the end of the lab, you should be able to:

- Code SELECT statements without or with conditions.
- Code SELECT statements using keywords BETWEEN, IN, LIKE and DISTINCT.
- Produce a result table ordered in desired order.

Introduction

See the data model at the start of this exercise guide to get the column names and descriptions for each table.

Required Materials

- Student handout
- SQL Reference

Exercises Instructions

Problem 1

List employee number, last name, date of birth, and salary for all employees. The employees should be ordered by decreasing salaries.

Problem 2

List last name, first name, and the department number for all employees. The listing should be ordered by descending department numbers. Within the same department, the last names should be sorted in descending order.

Problem 3

List the different education levels in the company in descending order. List only one occurrence of duplicate result rows.

Problem 4

List all employees (employee number) and the projects (project number) performed by them. List only one occurrence of duplicate result rows.

Use the EMP _ACT table.

Problem 5

List last name, salary, and bonus of all male employees.

Problem 6

List last name, salary, and commission of all employees with a salary higher than $20,000 and hired after 1979.

Problem 7

List last name, salary, bonus, and commission of all employees with a salary higher than $22,000 and a bonus of $400 or of all employees with a bonus of $500 and a commission lower than $1,900. The list should be ordered by last names.

Problem 8

List last name, salary, bonus, and commission of all employees with a salary higher than $22,000, a bonus of $400 or $500, and a commission lower than $1,900. The list should be ordered by the last names.

Problem 9

For all 'AD' projects (project number starts with 'AD') with activities 10, 80, and 180, list the following:

- Project number
- Activity number
- Starting date for activity
- Ending date for activity

The list should be sorted by project number and activity number.

Problem 10

List manager number and department number for all departments to which a manager has been assigned.

The list should be ordered by manager numbers.

Problem 11

List employee number, last name, salary, and bonus of all employees having a bonus from $800 to $1,000.

The lowest bonus should appear first.

Problem 12

List employee number, last name, salary, and department number of all employees of departments A00 through C01.

The listing should be ordered by ascending last names.

Problem 13

List all projects (ordered by the project number) for which 'SUPPORT' is a part of their name.

Problem 14

List all departments with a '1' as middle character of their department number.

Order the list by department numbers.

END OF LAB

Exercise Instructions With Hints (Expected Results)

Problem 1

List employee number, last name, date of birth, and salary for all employees. The employees should be ordered by decreasing salaries.

Result

```
EMPNO  LASTNAME        BIRTHDATE      SALARY
------ --------------- ---------- -----------
000010 HAAS            24/08/1933    52750,00
000110 LUCCHESSI       05/11/1929    46500,00
000020 THOMPSON        02/02/1948    41250,00
000050 GEYER           15/09/1925    40175,00
000030 KWAN            11/05/1941    38250,00
000070 PULASKI         26/05/1953    36170,00
000060 STERN           07/07/1945    32250,00

.....  ..... ....     ..........    ..........

The complete result set contains 32 rows.
```

Problem 2

List last name, first name, and the department number for all employees. The listing should be ordered by descending department numbers. Within the same department, the last names should be sorted in descending order.

Result

```
LASTNAME        FIRSTNME     WORKDEPT
--------------- ------------ --------

SPENSER         THEODORE       E21
MEHTA           RAMLAL         E21
LEE             WING           E21
GOUNOT          JASON          E21
SMITH           PHILIP         E11
SETRIGHT        MAUDE          E11
SCHNEIDER       ETHEL          E11
PARKER          JOHN           E11
HENDERSON       EILEEN         E11
GEYER           JOHN           E01
.....           .....          .....

The complete result set contains 32 rows.
```

Problem 3

List the different education levels in the company in descending order. List only one occurrence of duplicate result rows.

Result

```
EDLEVEL
-------
     20
     19
     18
     17
     16
     15
     14
     12
```

Problem 4

List all employees (employee number) and the projects (project number) performed by them. List only one occurrence of duplicate result rows.

Use the EMP _ACT table.

Result

```
EMPNO  PROJNO
------ ------
000010 AD3100
000070 AD3110
000230 AD3111
000240 AD3111
000250 AD3112
000260 AD3113
000270 AD3113
000030 IF1000
000130 IF1000
000140 IF1000
000030 IF2000
000140 IF2000
...... ......

The complete result set contains 36 rows.
```

Problem 5

List last name, salary, and bonus of all male employees.

Result

```
LASTNAME        SALARY        BONUS
--------------- ------------ --------
THOMPSON        41250,00       800,00
GEYER           40175,00       800,00
STERN           32250,00       500,00
SPENSER         26150,00       500,00
LUCCHESSI       46500,00       900,00
O'CONNELL       29250,00       600,00
ADAMSON         25280,00       500,00
YOSHIMURA       24680,00       500,00
WALKER          20450,00       400,00
.............    .............  .........

The complete result set contains 19 rows.
```

Problem 6

List last name, salary, and commission of all employees with a salary higher than $20,000 and hired after 1979.

Result

```
LASTNAME        SALARY       COMM
--------------- ------------ ---------
PULASKI         36170,00     2893,00
SPENSER         26150,00     2092,00
PEREZ           27380,00     2190,00
```

Problem 7

List last name, salary, bonus, and commission of all employees with a salary higher than $22,000 and a bonus of $400 or of all employees with a bonus of $500 and a commission lower than $1,900. The list should be ordered by last names.

Result

```
LASTNAME        SALARY       BONUS        COMM
--------------- ------------ ------------ ---------
JEFFERSON       22180,00     400,00       1774,00
PIANKA          22250,00     400,00       1780,00
SCOUTTEN        21340,00     500,00       1707,00
```

Problem 8

List last name, salary, bonus, and commission of all employees with a salary higher than $22,000, a bonus of $400 or $500, and a commission lower than $1,900. The list should be ordered by the last names.

Result

```
LASTNAME        SALARY      BONUS       COMM
--------------- ----------- ----------- ---------
JEFFERSON       22180,00    400,00      1774,00
PIANKA          22250,00    400,00      1780,00
```

Problem 9

For all 'AD' projects (project number starts with 'AD') with activities 10, 80, and 180, list the following:

- Project number
- Activity number
- Starting date for activity
- Ending date for activity

The list should be sorted by project number and activity number.

Result

```
PROJNO ACTNO  EMSTDATE   EMENDATE
------ ------ ---------- ----------
AD3100    10  01/01/1982 01/07/1982
AD3110    10  01/01/1982 01/02/1983
AD3111    80  15/04/1982 15/10/1982
AD3111    80  15/09/1982 01/01/1983
AD3111   180  15/10/1982 01/01/1983
AD3112    80  15/08/1982 15/10/1982
AD3112    80  15/10/1982 01/12/1982
AD3112   180  15/08/1982 01/01/1983
AD3113    80  01/01/1982 01/03/1982
AD3113    80  01/03/1982 15/04/1982
AD3113    80  01/01/1982 01/03/1982
AD3113    80  01/03/1982 01/04/1982
AD3113   180  01/03/1982 15/04/1982
AD3113   180  15/04/1982 01/06/1982
AD3113   180  01/06/1982 01/07/1982
```

Problem 10

List manager number and department number for all departments to which a manager has been assigned.

The list should be ordered by manager numbers.

Result

```
MGRNO  DEPTNO
------ ------
000010  A00
000020  B01
000030  C01
000050  E01
000060  D11
000070  D21
000090  E11
000100  E21
```

Problem 11

List employee number, last name, salary, and bonus of all employees having a bonus from $800 to $1,000.

The lowest bonus should appear first.

Result

```
EMPNO  LASTNAME        SALARY      BONUS
------ --------------- ----------- ---------
000020 THOMPSON        41250,00     800,00
000030 KWAN            38250,00     800,00
000050 GEYER           40175,00     800,00
000110 LUCCHESSI       46500,00     900,00
000010 HAAS            52750,00    1000,00
```

Problem 12

List employee number, last name, salary, and department number of all employees of departments A00 through C01.

The listing should be ordered by ascending last names.

Result

```
EMPNO  LASTNAME        SALARY      WORKDEPT
------ --------------- ----------- --------
000010 HAAS            52750,00     A00
000030 KWAN            38250,00     C01
000110 LUCCHESSI       46500,00     A00
000140 NICHOLLS        28420,00     C01
000120 O'CONNELL       29250,00     A00
000130 QUINTANA        23800,00     C01
000020 THOMPSON        41250,00     B01
```

Problem 13

List all projects (ordered by the project number) for which 'SUPPORT' is a part of their name.

Result

```
PROJNO PROJNAME
------ ------------------------
OP1000 OPERATION SUPPORT
OP2010 SYSTEMS SUPPORT
OP2011 SCP SYSTEMS SUPPORT
OP2012 APPLICATIONS SUPPORT
OP2013 DB/DC SUPPORT
```

Problem 14

List all departments with a '1' as middle character of their department number.

Order the list by department numbers.

Result

```
DEPTNO DEPTNAME
------ ----------------------------
D11    MANUFACTURING SYSTEMS
E11    OPERATIONS
```

END OF LAB (with hints)

Solution

Problem 1

```
SELECT EMPNO, LASTNAME, BIRTHDATE, SALARY
   FROM EMPLOYEE
 ORDER BY SALARY DESC
```

Problem 2

```
SELECT LASTNAME, FIRSTNME, WORKDEPT
   FROM EMPLOYEE
 ORDER BY WORKDEPT DESC, LASTNAME DESC
```

Problem 3

```
SELECT DISTINCT EDLEVEL
   FROM EMPLOYEE
 ORDER BY EDLEVEL DESC
```

Problem 4

```
SELECT DISTINCT EMPNO, PROJNO
   FROM EMP_ACT
```

Problem 5

```
SELECT LASTNAME, SALARY, BONUS
   FROM EMPLOYEE
 WHERE SEX = 'M'
```

Problem 6

```
SELECT LASTNAME, SALARY, COMM
   FROM EMPLOYEE
 WHERE HIREDATE >= '1980-01-01'
    AND SALARY > 20000
```

Problem 7

```
SELECT LASTNAME, SALARY, BONUS, COMM
   FROM EMPLOYEE
 WHERE SALARY > 22000 AND BONUS = 400
    OR BONUS = 500 AND COMM < 1900
 ORDER BY LASTNAME
```

Problem 8

```
SELECT LASTNAME, SALARY, BONUS, COMM
   FROM EMPLOYEE
 WHERE SALARY > 22000
    AND (BONUS = 400 OR BONUS = 500)
    AND COMM < 1900
 ORDER BY LASTNAME
```

Problem 9

```
SELECT PROJNO, ACTNO, EMSTDATE, EMENDATE
   FROM EMP_ACT
 WHERE ACTNO IN (10, 80, 180)
    AND   PROJNO LIKE 'AD%'
 ORDER BY PROJNO, ACTNO
```

Problem 10

```
SELECT MGRNO, DEPTNO
   FROM DEPARTMENT
 WHERE MGRNO IS NOT NULL
 ORDER BY MGRNO
```

Problem 11

```
SELECT EMPNO, LASTNAME, SALARY, BONUS
   FROM EMPLOYEE
 WHERE BONUS BETWEEN 800 AND 1000
 ORDER BY BONUS
```

Problem 12

```
SELECT EMPNO, LASTNAME, SALARY, WORKDEPT
   FROM EMPLOYEE
 WHERE WORKDEPT BETWEEN 'A00' AND 'C01'
 ORDER BY LASTNAME ASC
```

Problem 13

```
SELECT PROJNO, PROJNAME
   FROM PROJECT
 WHERE PROJNAME LIKE '%SUPPORT%'
 ORDER BY PROJNO
```

Problem 14

```
SELECT DEPTNO, DEPTNAME
   FROM DEPARTMENT
 WHERE DEPTNO LIKE '_1_'
 ORDER BY DEPTNO
```

END OF SOLUTION

Exercise 5. Retrieving Data from Multiple Tables

What This Exercise is About

This exercise provides an opportunity to write joins.

What You Should Be Able to Do

At the end of the lab, you should be able to:

- Code SELECT statements referencing multiple tables or views.

Introduction

See the data model at the start of this exercise guide to get the column names and descriptions for each table.

Required Materials

- Student handout
- SQL Reference

Exercises Instructions

Problem 1

For all projects (project number) beginning with 'AD', list project number, project name, and activity number. List identical rows once.

Order the list by project number and activity number.

Problem 2

Which employees were assigned to project 'AD3113'? List employee number, last name, and project number. Order the list by employee number and project number. List only one occurrence of duplicate result rows.

Problem 3

Which activities started on '1982-10-01'? For each of these activities, list employee number of the person performing the activity, project number, project name, activity number, and starting date of the activity. Order the list by project number, employee number, and activity number.

Problem 4

List department number, manager number, and last name of the managers of departments 'A00' and 'B01'.

Sort the list by department number.

Problem 5

List department number, last name, project name, and activity number for activities performed by the employees of department 'A00'.

The list should be sorted by project name and activity number.

Problem 6

List department number, last name, project name, and activity number for all employees. Suppress identical rows.

Sort the list by department number, last name, and activity number.

Problem 7

Our second line manager needs a list of activities which started on 1982-10-15 or thereafter.

For these activities, list activity number, manager number responsible for the department performing the project, the starting date for the activity, project number, and last name of the employee performing the activity.

The list should be ordered by activity number and activity starting date.

Problem 8

Which employees of department 'A00' were hired before their manager?

List department number, manager's last name, employee's last name, hiring dates of the manager and employee.

Order the list by employee last name.

END OF LAB

Exercise Instructions With Hints (Expected Results)

Problem 1

For all projects (project number) beginning with 'AD', list project number, project name, and activity number. List identical rows once.

Order the list by project number and activity number.

Result

```
PROJNO PROJNAME                   ACTNO
------ ------------------------ ------
AD3100 ADMIN SERVICES                10
AD3110 GENERAL ADMIN SYSTEMS         10
AD3111 PAYROLL PROGRAMMING           60
AD3111 PAYROLL PROGRAMMING           70
AD3111 PAYROLL PROGRAMMING           80
AD3111 PAYROLL PROGRAMMING          180
AD3112 PERSONNEL PROGRAMMING         60
AD3112 PERSONNEL PROGRAMMING         70
AD3112 PERSONNEL PROGRAMMING         80
AD3112 PERSONNEL PROGRAMMING        180
AD3113 ACCOUNT PROGRAMMING           60
AD3113 ACCOUNT PROGRAMMING           70
AD3113 ACCOUNT PROGRAMMING           80
AD3113 ACCOUNT PROGRAMMING          180
```

Problem 2

Which employees were assigned to project 'AD3113'? List employee number, last name, and project number. Order the list by employee number and project number. List only one occurrence of duplicate result rows.

Result

```
EMPNO  LASTNAME        PROJNO
------ --------------- ------
000260 JOHNSON         AD3113
000270 PEREZ           AD3113
```

Problem 3

Which activities started on '1982-10-01'? For each of these activities, list employee number of the person performing the activity, project number, project name, activity number, and starting date of the activity. Order the list by project number, employee number, and activity number.

Result

```
EMPNO  PROJNO PROJNAME                  ACTNO  EMSTDATE
------ ------ ------------------------ ------ ----------
000130 IF1000 QUERY SERVICES              100   01/10/1982
000140 IF1000 QUERY SERVICES               90   01/10/1982
000140 IF2000 USER EDUCATION              110   01/10/1982
000190 MA2112 W L ROBOT DESIGN             80   01/10/1982
000210 MA2113 W L PROD CONT PROGS          80   01/10/1982
000210 MA2113 W L PROD CONT PROGS         180   01/10/1982
```

Problem 4

List department number, manager number, and last name of the managers of departments 'A00' and 'B01'.

Sort the list by department number.

Result

```
DEPTNO MGRNO  LASTNAME
------ ------ ---------------
A00    000010 HAAS
B01    000020 THOMPSON
```

Problem 5

List department number, last name, project name, and activity number for activities performed by the employees of department 'A00'.

The list should be sorted by project name and activity number.

Result

```
WORKDEPT LASTNAME     PROJNAME                 ACTNO
-------- ------------ ------------------------ ------
A00      HAAS         ADMIN SERVICES             10
A00      HAAS         W L PROGRAMMING            10
A00      HAAS         WELD LINE AUTOMATION       10
A00      LUCCHESSI    WELD LINE AUTOMATION       20
```

Problem 6

List department number, last name, project name, and activity number for all employees. Suppress identical rows.

Sort the list by department number, last name, and activity number.

Result

```
WORKDEPT LASTNAME        PROJNAME                  ACTNO
-------- --------------- ------------------------ ------
A00      HAAS            ADMIN SERVICES               10
A00      HAAS            W L PROGRAMMING              10
A00      HAAS            WELD LINE AUTOMATION         10
A00      LUCCHESSI       WELD LINE AUTOMATION         20
B01      THOMPSON        WELD LINE PLANNING           30
C01      KWAN            QUERY SERVICES               10
C01      KWAN            USER EDUCATION               10
C01      NICHOLLS        QUERY SERVICES               90
C01      NICHOLLS        USER EDUCATION              100
C01      NICHOLLS        USER EDUCATION              110
C01      QUINTANA        QUERY SERVICES               90
C01      QUINTANA        QUERY SERVICES              100
D11      ADAMSON         W L ROBOT DESIGN             60
D11      ADAMSON         W L ROBOT DESIGN            180
D11      BROWN           W L PROGRAM DESIGN           50
D11      BROWN           W L PROGRAM DESIGN           60
D11      JONES           W L PROD CONT PROGS          80
D11      JONES           W L PROD CONT PROGS         180
...... ..... .... ..... ..... ..... ..... ..... .....

The complete result set contains 57 rows.
```

Problem 7

Our second line manager needs a list of activities which started on 1982-10-15 or thereafter.

For these activities, list activity number, manager number responsible for the department performing the project, the starting date for the activity, project number, and last name of the employee performing the activity.

The list should be ordered by activity number and activity starting date.

Result

```
ACTNO  MGRNO  EMSTDATE   PROJNO LASTNAME
------ ------ ---------- ------ ----------
    60 000070 01/12/1982 AD3112 SMITH
    60 000070 01/01/1983 AD3112 SMITH
    70 000070 15/10/1982 AD3113 PEREZ
    80 000070 15/10/1982 AD3112 SMITH
   180 000070 15/10/1982 AD3111 JEFFERSON
```

Problem 8

Which employees of department 'A00' were hired before their manager?

List department number, manager's last name, employee's last name, hiring dates of the manager and employee.

Order the list by employee last name.

Result

```
DEPTNO LASTNAME    LASTNAME       HIREDATE   HIREDATE
------ ----------------------------- ---------- --------
A00    HAAS        LUCCHESSI      01/01/1965 16/05/1958
A00    HAAS        O'CONNELL      01/01/1965 05/12/1963
```

END OF LAB (with hints)

Solution

Problem 1

```
SELECT DISTINCT A.PROJNO, PROJNAME, ACTNO
   FROM PROJECT P, EMP_ACT A
 WHERE P.PROJNO = A.PROJNO
    AND A.PROJNO LIKE 'AD%'
 ORDER BY A.PROJNO, ACTNO
```

Problem 2

```
SELECT DISTINCT A.EMPNO, LASTNAME, PROJNO
   FROM EMPLOYEE E, EMP_ACT A
 WHERE A.EMPNO = E.EMPNO
    AND A.PROJNO = 'AD3113'
 ORDER BY A.EMPNO, PROJNO
```

Problem 3

```
SELECT A.EMPNO, A.PROJNO, PROJNAME, ACTNO, EMSTDATE
   FROM PROJECT P, EMP_ACT A
 WHERE A.PROJNO = P.PROJNO
    AND EMSTDATE = '1982-10-01'
 ORDER BY A.PROJNO, A.EMPNO, ACTNO
```

Problem 4

```
SELECT DEPTNO, MGRNO, LASTNAME
   FROM EMPLOYEE E, DEPARTMENT D
 WHERE E.EMPNO=D.MGRNO
    AND DEPTNO IN ('A00','B01')
 ORDER BY DEPTNO
```

Problem 5

```
SELECT WORKDEPT, LASTNAME, PROJNAME, ACTNO
   FROM EMPLOYEE E, PROJECT P, EMP_ACT A
 WHERE E.EMPNO = A.EMPNO
    AND A.PROJNO = P.PROJNO
    AND WORKDEPT='A00'
 ORDER BY PROJNAME, ACTNO
```

Problem 6

```
SELECT DISTINCT WORKDEPT, LASTNAME, PROJNAME, ACTNO
   FROM EMPLOYEE E, PROJECT P, EMP_ACT A
 WHERE E.EMPNO = A.EMPNO
    AND A.PROJNO = P.PROJNO
 ORDER BY WORKDEPT, LASTNAME, ACTNO
```

Problem 7

```
SELECT ACTNO, D.MGRNO, EMSTDATE, P.PROJNO, LASTNAME
  FROM DEPARTMENT D, EMPLOYEE E, PROJECT P, EMP_ACT A
 WHERE E.EMPNO = A.EMPNO
   AND A.PROJNO = P.PROJNO
   AND E.WORKDEPT = D.DEPTNO
AND A.EMSTDATE >= '1982-10-15'
ORDER BY ACTNO, EMSTDATE
```

Problem 8

```
SELECT DEPTNO, M.LASTNAME, E.LASTNAME, M.HIREDATE,
E.HIREDATE
   FROM EMPLOYEE M, EMPLOYEE E, DEPARTMENT D
 WHERE D.MGRNO = M.EMPNO
    AND E.WORKDEPT = D.DEPTNO
    AND M.HIREDATE > E.HIREDATE
    AND E.WORKDEPT = 'A00'
 ORDER BY E.LASTNAME
```

END OF SOLUTION

Exercise 6. Scalar Functions and Arithmetic

What This Exercise is About

This exercise provides an opportunity to work with scalar functions and arithmetic expressions.

What You Should Be Able to Do

At the end of the lab, you should be able to:

- Code queries that use scalar functions
- Code queries by using calculated expressions in the select list and in the WHERE clause

Introduction

See the data model at the start of this exercise guide to get the column names and descriptions for each table.

Required Materials

- Student handout
- SQL Reference

Exercises Instructions

Problem 1

For employees whose salary, in creased by 5 percent, is less than or equal to $20,000, list the following:

- Last name
- Salary
- Salary increased by 5 percent
- Monthly salary increased by 5 percent

Sort the listing by salary.

Problem 2

For all employees with an education level of 18 or 20, the salary will be increased by $1,200, and the bonus will be halved. List last name, education level, the new salary and new bonus for these employees.

Employees with an education level of 20 should be listed first. For employees with same education level, sort the list by salary.

Problem 3

The salary will be decreased by $1,000 for all employees matching the following criteria:

- They belong to department 'D11'.
- Their salary is more than or equal to 80 percent of $20.000.
- Their salary is less than or equal to 120 percent of $20,000.

List department number, last name, salary, and decreased salary. Sort the result by salary.

Problem 4

We need a list for all employees of department 'D11' with an income (sum of salary, commission, and bonus) that is higher than their salary increased by 10 percent. List department number, last name, and income.

Sort the result in descending order by income.

Problem 5

List all departments which have no manager assigned. List department number and department name.

Problem 6

List project number and major project number of all projects whose project number contains 'MA' in position 1 and 2. If the major project number is unknown, display the text 'MAI N PROJECT'.

Problem 7

List all employees who were younger than 25 when they joined the company.

List their employee number, last name, and age in years.

Sort the result by age of employee when joining the company and employee number.

Problem 8

We need a list of all projects which ended on 1982-12-01. Display year and month of the estimated starting date and project number. Sort the result by the project number.

Problem 9

List project number and duration of all 'MA' projects (project number start with 'MA') in weeks. The duration should be rounded and displayed with one decimal position.

Order the list by the project number.

Problem 10

For projects whose project number starts with 'MA', list project number, estimated ending dates, and expected ending dates, if the duration will exceed the estimated by 10 percent.

Order the list by project number.

Problem 11

How many days are there until the end of this year? (The expected result was produced on 1999-04-21 and, therefore, will not match your result).

You may use an arbitrary table. Specify a predicate selecting a single row.

For OS/390 you may use SYSIBM.SYSDUMMY1 table which contains a single row.

Problem 12

List the first name initial and last name of all employees whose last name starts with 'S'. The initial should be separated from the last name by means of a period.

Problem 13

For all employees with an employee number from '000010' to '000130', display the following data, thereby, replacing some values by a more readable text:

- Employee number
- The first two characters of the project number replace by the following text (use column name PROJNO and name it PROJ):

  AD = ADMIN.
  IF = SUP.
  MA = DEV.
  OP = OPER.
  PL = PLAN.
- Activity number replace by the following text (use column name ACTNO and name it ACT):

  10 = MANAGE
  20 = COST ESTIMATE
  30 = DEFINE SPECS

  If activity number is not 10, 20, or 30 display 'OTHER'.
- Activity starting date
- Activity ending date

Display only those activities which ended before 1989-10-01.

Sort the result by employee number, PROJ column, activity starting date, ACT column, and activity ending date.

END OF LAB

Exercise Instructions With Hints (Expected Results)

Problem 1

For employees whose salary, in creased by 5 percent, is less than or equal to $20,000, list the following:

- Last name
- Salary
- Salary increased by 5 percent
- Monthly salary increased by 5 percent

Sort the listing by salary.

Result

```
LASTNAME      SALARY       INC-Y-SALARY    INC-M-SALARY
--------------------- -------------- ------------
PARKER        15340,00      16107,0000     1342,2500000
SETRIGHT      15900,00      16695,0000     1391,2500000
JOHNSON       17250,00      18112,5000     1509,3750000
SMITH         17750,00      18637,5000     1553,1250000
JONES         18270,00      19183,5000     1598,6250000
```

Problem 2

For all employees with an education level of 18 or 20, the salary will be increased by $1,200, and the bonus will be halved. List last name, education level, the new salary and new bonus for these employees.

Employees with an education level of 20 should be listed first. For employees with same education level, sort the list by salary.

Result

```
LASTNAME        EDLEVEL     NEW-SALARY  NEW-BONUS
--------------- ------- ---------------- ----------
KWAN                 20       39450,00     400,00
NICHOLLS             18       29620,00     300,00
LUTZ                 18       31040,00     300,00
THOMPSON             18       42450,00     400,00
HAAS                 18       53950,00     500,00
```

Problem 3

The salary will be decreased by $1,000 for all employees matching the following criteria:

- They belong to department 'D11'.
- Their salary is more than or equal to 80 percent of $20.000.
- Their salary is less than or equal to 120 percent of $20,000.

List department number, last name, salary, and decreased salary. Sort the result by salary.

Result

```
WORKDEPT LASTNAME        SALARY       DECR-SALARY
-------- --------------- ------------ -----------
D11      JONES           18270,00       17270,00
D11      WALKER          20450,00       19450,00
D11      SCOUTTEN        21340,00       20340,00
D11      PIANKA          22250,00       21250,00
```

Problem 4

We need a list for all employees of department 'D11' with an income (sum of salary, commission, and bonus) that is higher than their salary increased by 10 percent. List department number, last name, and income.

Sort the result in descending order by income.

Result

```
WORKDEPT LASTNAME        INCOME
-------- --------------- --------
D11      LUTZ            32827,00
D11      BROWN           30557,00
D11      YOSHIMURA       27154,00
D11      SCOUTTEN        23547,00
D11      JONES           20132,00
```

Problem 5

List all departments which have no manager assigned. List department number and department name.

Result

```
DEPTNO  DEPTNAME
------ ------------------
D01     DEVELOPMENT CENTER
```

Problem 6

List project number and major project number of all projects whose project number contains 'MA' in position 1 and 2. If the major project number is unknown, display the text 'MAI N PROJECT'.

Result

```
PROJNO      MAJOR PROJECT
------      -------------
MA2100      MAIN PROJECT
MA2110      MA2100
MA2111      MA2110
MA2112      MA2110
MA2113      MA2110
```

Problem 7

List all employees who were younger than 25 when they joined the company.

List their employee number, last name, and age in years.

Sort the result by age of employee when joining the company and employee number.

Result

```
EMPNO  LASTNAME        AGE
------ --------------- -----
000220 LUTZ             20
000340 GOUNOT           20
000120 O'CONNELL        21
000160 PIANKA           22
000190 WALKER           22
000050 GEYER            23
000100 SPENSER          23
000150 ADAMSON          24
000180 SCOUTTEN         24
000200 BROWN            24
```

Problem 8

We need a list of all projects which ended on 1982-12-01. Display year and month of the estimated starting date and project number. Sort the result by the project number.

Result

```
YEAR        MONTH     PROJNO
----------- --------- ------
1982          1       MA2111
1982          1       A2112
1982          2        MA2113
```

Problem 9

List project number and duration of all 'MA' projects (project number start with 'MA') in weeks. The duration should be rounded and displayed with one decimal position.

Order the list by the project number.

Result

```
PROJNO       WEEKS
------ -----------
MA2100        56,6
MA2110        56,6
MA2111        47,7
MA2112        47,7
MA2113        41,3
```

Problem 10

For projects whose project number starts with 'MA', list project number, estimated ending dates, and expected ending dates, if the duration will exceed the estimated by 10 percent.

Order the list by project number.

Result

```
PROJNO ESTIMATED  EXPECTED
------ ---------- ----------
MA2100 01/02/1983 12/03/1983
MA2110 01/02/1983 12/03/1983
MA2111 01/12/1982 03/01/1983
MA2112 01/12/1982 03/01/1983
MA2113 01/12/1982 29/12/1982
```

Problem 11

How many days are there until the end of this year? (The expected result was produced on 2003-09-19 and, therefore, will not match your result).

You may use an arbitrary table. Specify a predicate selecting a single row.

For OS/390 you may use SYSIBM.SYSDUMMY1 table which contains a single row.

Result

```
DAYS
------
103
```

Problem 12

List the first name initial and last name of all employees whose last name starts with 'S'. The initial should be separated from the last name by means of a period.

Result

```
NAME
----------------
I.STERN
T.SPENSER
M.SCOUTTEN
D.SMITH
E.SCHNEIDER
P.SMITH
M.SETRIGHT
```

Problem 13

For all employees with an employee number from '000010' to '000130', display the following data, thereby, replacing some values by a more readable text:

- Employee number
- The first two characters of the project number replace by the following text (use column name PROJNO and name it PROJ):

  AD = ADMIN.
  IF = SUP.
  MA = DEV.
  OP = OPER.
  PL = PLAN.
- Activity number replace by the following text (use column name ACTNO and name it ACT):

  10 = MANAGE
  20 = COST ESTIMATE
  30 = DEFINE SPECS

  If activity number is not 10, 20, or 30 display 'OTHER'.
- Activity starting date
- Activity ending date

Display only those activities which ended before 1989-10-01.

Sort the result by employee number, PROJ column, activity starting date, ACT column, and activity ending date.

Result

EMPNO	PROJ	ACT	EMSTDATE	EMENDATE
000010	ADMIN	MANAGER	01/01/1982	01/07/1982
000010	DEV	MANAGER	01/01/1982	01/11/1982
000010	DEV	MANAGER	01/01/1982	01/02/1983
000020	PLAN	DEFINE SPECS	01/01/1982	15/09/1982
000030	SUP	MANAGER	01/01/1982	01/01/1983
000030	SUP	MANAGER	01/06/1982	01/01/1983
000050	OPER	MANAGER	01/01/1982	01/02/1983
000070	ADMIN	MANAGER	01/01/1982	01/02/1983
000090	OPER	MANAGER	01/01/1982	01/02/1983
000100	OPER	MANAGER	01/01/1982	01/02/1983
000110	DEV	COST ESTIMATE	01/01/1982	01/03/1982
000130	SUP	OTHER	01/01/1982	01/10/1982
000130	SUP	OTHER	01/10/1982	01/01/1983

END OF LAB (with hints)

Solution

Problem 1

```
SELECT LASTNAME, SALARY,
                 SALARY*1.05     AS "INC-Y-SALARY",
                 SALARY*1.05/12. AS "INC-M-SALARY"
   FROM EMPLOYEE
 WHERE SALARY*1.05 <= 20000
 ORDER BY SALARY
```

Problem 2

```
SELECT LASTNAME, EDLEVEL,
        SALARY+1200 AS "NEW-SALARY",
        BONUS*0.5 AS "NEW-BONUS"
   FROM EMPLOYEE
 WHERE EDLEVEL = 18 OR EDLEVEL = 20
 ORDER BY EDLEVEL DESC, 3
```

Problem 3

```
SELECT WORKDEPT, LASTNAME, SALARY,
        SALARY-1000 AS "DECR-SALARY"
   FROM EMPLOYEE
 WHERE SALARY BETWEEN 20000*0.8 AND 20000*1.20
    AND WORKDEPT = 'D11'
 ORDER BY SALARY
```

Problem 4

```
SELECT WORKDEPT, LASTNAME, SALARY+COMM+BONUS AS INCOME
   FROM EMPLOYEE
 WHERE SALARY+COMM+BONUS > 1.1*SALARY
   AND WORKDEPT = 'D11'
 ORDER BY 3 DESC
```

Problem 5

```
SELECT DEPTNO, DEPTNAME
   FROM DEPARTMENT
 WHERE MGRNO IS NULL
```

Problem 6

```
SELECT PROJNO,
  COALESCE(MAJPROJ,'MAIN PROJECT') AS "MAJOR PROJECT"
   FROM PROJECT
 WHERE SUBSTR (PROJNO, 1, 2) = 'MA'
```

Problem 7

```
SELECT EMPNO, LASTNAME, YEAR(HIREDATE-BIRTHDATE)AS AGE
   FROM EMPLOYEE
 WHERE YEAR(HIREDATE-BIRTHDATE) < 25
 ORDER BY 3, EMPNO
```

Problem 8

```
SELECT YEAR(PRSTDATE) AS YEAR, MONTH(PRSTDATE) AS
       MONTH, PROJNO
   FROM PROJECT
 WHERE PRENDATE = '1982-12-01'
 ORDER BY PROJNO
```

Problem 9

```
SELECT PROJNO, DECIMAL((DAYS(PRENDATE) -
                DAYS(PRSTDATE))/7.+0.05, 8, 1) AS WEEK
   FROM PROJECT
 WHERE PROJNO LIKE 'MA%'
 ORDER BY PROJNO
```

Problem 10

```
SELECT PROJNO, PRENDATE AS ESTIMATED,
       PRSTDATE+((DAYS(PRENDATE)-DAYS(PRSTDATE))*1.1)
       DAYS AS EXPECTED
   FROM PROJECT
 WHERE PROJNO LIKE 'MA%'
 ORDER BY PROJNO
```

Problem 11

```
SELECT DAYS('2003-12-31')-DAYS('2003-09-19') AS DAYS
FROM SYSIBM.SYSDUMMY1
```

Problem 12

```
SELECT SUBSTR(FIRSTNME, 1, 1) CONCAT  '.'  CONCAT
       LASTNAME AS NAME
   FROM EMPLOYEE
 WHERE LASTNAME LIKE 'S%'
```

Problem 13

```
SELECT DISTINCT EMPNO,
      CASE SUBSTR(PROJNO, 1, 2)
      WHEN 'AD' THEN 'ADMIN'
      WHEN 'IF' THEN 'SUP'
      WHEN 'MA' THEN 'DEV'
      WHEN 'OP' THEN 'OPER'
      WHEN 'PL' THEN 'PLAN'
      END AS PROJ,
      CASE
      WHEN ACTNO = 10 THEN 'MANAGER'
      WHEN ACTNO = 20 THEN 'COST ESTIMATE'
      WHEN ACTNO = 30 THEN 'DEFINE SPECS'
      ELSE 'OTHER'
      END AS ACT,
      EMSTDATE, EMENDATE
   FROM EMP_ACT
 WHERE EMENDATE<='1989-10-01'
    AND EMPNO BETWEEN '000010' AND '000130'
 ORDER BY EMPNO, PROJ, EMSTDATE, ACT, EMENDATE
```

END OF SOLUTION

Exercise 7. Column Functions and Grouping

What This Exercise is About

This exercise provides an opportunity to work with column functions.

What You Should Be Able to Do

At the end of the lab, you should be able to:

- Code queries that use column functions

Introduction

See the data model at the start of this exercise guide to get the column names and descriptions for each table.

Required Materials

- Student handout
- SQL Reference

Exercises Instructions

Problem 1

For all departments, display department number and the sum of all salaries for each department.

Problem 2

For all departments, display the department number and the number of employees.

Problem 3

Display the departments which have more than 3 employees.

Problem 4

For all departments with at least one designer, display the number of designers and the department number.

Problem 5

For each department and each sex, display the average salary, the average bonus, the average commission, and number of persons if the sex group contains at least two persons. The average should be truncated (no rounding) after two decimal positions.

Problem 6

Display average bonus and average commission for all departments with an average bonus higher than $500 and an average commission higher than $2,000.

END OF LAB

Exercise Instructions With Hints (Expected Results)

Problem 1

For all departments, display department number and the sum of all salaries for each department.

Result

```
WORKDEPT SUM_SALARY
-------- ----------
A00      128500,00
B01       41250,00
C01       90470,00
D11      222100,00
D21      150920,00
E01       40175,00
E11      104990,00
E21       95310,00
```

Problem 2

For all departments, display the department number and the number of employees.

Result

```
WORKDEPT   EMP_COUNT
-------- -----------
A00              3
B01              1
C01              3
D11              9
D21              6
E01              1
E11              5
E21              4
```

Problem 3

Display the departments which have more than 3 employees.

Result

```
WORKDEPT
--------
D11
D21
E11
E21
```

Problem 4

For all departments with at least one designer, display the number of designers and the department number.

Result

```
DESIGNER    WORKDEPT
----------- --------
   8          D11
```

Problem 5

For each department and each sex, display the average salary, the average bonus, the average commission, and number of persons if the sex group contains at least two persons. The average should be truncated (no rounding) after two decimal positions.

Result

```
WORKDEPT SEX AVG_SALARY  AVG_BONUS   AVG_COMM    COUNT
-------- --- ---------- ---------- ---------- ------
A00      M      37875,00     750,00    3030,00      2
C01      F      30156,66     633,33    2412,66      3
D11      F      24476,66     500,00    1958,00      3
D11      M      24778,33     483,33    1981,83      6
D21      F      26933,33     500,00    2154,33      3
D21      M      23373,33     466,66    1869,66      3
E11      F      23966,66     466,66    1917,33      3
E11      M      16545,00     350,00    1323,50      2
E21      M      23827,50     475,00    1906,25      4
```

Problem 6

Display average bonus and average commission for all departments with an average bonus higher than $500 and an average commission higher than $2,000.

Result

```
WORKDEPT        AVG-BONUS          AVG-COMM
-------- -------------------- -----------
A00           833,3333333     3426,6666666
B01           800,0000000     3300,0000000
C01           633,3333333     2412,6666666
E01           800,0000000     3214,0000000
```

END OF LAB (with hints)

Solution

Problem 1

```
SELECT WORKDEPT, SUM(SALARY) AS SUM_SALARY
 FROM EMPLOYEE
 GROUP BY WORKDEPT
```

Problem 2

```
SELECT WORKDEPT, COUNT(*) AS EMP_COUNT
   FROM EMPLOYEE
 GROUP BY WORKDEPT
```

Problem 3

```
SELECT WORKDEPT
   FROM EMPLOYEE
 GROUP BY WORKDEPT
HAVING COUNT(*) > 3
```

Problem 4

```
SELECT COUNT(*) AS DESIGNER, WORKDEPT
   FROM EMPLOYEE
 WHERE JOB = 'DESIGNER'
 GROUP BY WORKDEPT
```

Problem 5

```
SELECT WORKDEPT, SEX, DECIMAL(AVG(SALARY), 8, 2)
                      AS "AVG_SALARY",
                      DECIMAL(AVG(BONUS), 8, 2)
                      AS "AVG_BONUS",
                      DECIMAL(AVG(COMM), 8, 2)
                      AS "AVG_COMM",
                      COUNT(*) AS COUNT
   FROM EMPLOYEE
 GROUP BY WORKDEPT, SEX
HAVING COUNT(*) > 1
```

Problem 6

```
SELECT WORKDEPT, AVG(BONUS) AS "AVG-BONUS",
                 AVG(COMM)  AS "AVG-COMM"
   FROM EMPLOYEE
 GROUP BY WORKDEPT
HAVING AVG(BONUS) > 500 AND AVG(COMM) > 2000
```

END OF SOLUTION

Exercise 8. UNION

What This Exercise is About

This exercise provides an opportunity to work with keyword UNION.

What You Should Be Able to Do

At the end of the lab, you should be able to:

- Derive result tables by combining multiple result tables.

Introduction

See the data model at the start of this exercise guide to get the column names and descriptions for each table.

Required Materials

- Student handout
- SQL Reference

Exercises Instructions

Problem 1

List the salaries of all employees in department 'A00'. For the last line of the report, display the sum of all salaries.

Problem 2

For departments A00, B01, and C01, list the projects they perform and the employees of the department. The output should consist of up to three lines for each department as follows:

First line

- Department number
- Text: DEPARTMENT
- Department name

Second line (if data available):

- Department number
- Project number
- Project name

Subsequent line (if data available):

- Department number
- Employee number
- Last name

Problem 3

For all projects (project number) beginning with 'IF', list the following:

First line:

- Text: PROJECT
- Project number
- Employee number responsible for the project
- Estimated starting date
- Estimated ending date

Subsequent line:

- Project number
- Employee number performing the activity
- Activity starting date
- Activity ending date

Display the projects in ascending sequence of their project number and employee number.

END OF LAB

Exercise Instructions With Hints (Expected Results)

Problem 1

List the salaries of all employees in department 'A00'. For the last line of the report, display the sum of all salaries.

Result

WORKDEPT	EMPNO	SALARY
A00	000120	29250,00
A00	000110	46500,00
A00	000010	52750,00
A00	SUM	128500,00

Problem 2

For departments A00, B01, and C01, list the projects they perform and the employees of the department. The output should consist of up to three lines for each department as follows:

First line

- Department number
- Text: DEPARTMENT
- Department name

Second line (if data available):

- Department number
- Project number
- Project name

Subsequent line (if data available):

- Department number
- Employee number
- Last name

Result

```
DEPTNO INFO       DEPTNAME                          4
------ ---------- ------------------------------ -----
A00    DEPARTMENT SPIFFY COMPUTER SERVICE DIV.       1
A00    000010     HAAS                               3
A00    000110     LUCCHESSI                          3
A00    000120     O'CONNELL                          3
B01    DEPARTMENT PLANNING                           1
B01    PL2100     WELD LINE PLANNING                 2
B01    000020     THOMPSON                           3
C01    DEPARTMENT INFORMATION CENTER                 1
C01    IF1000     QUERY SERVICES                     2
C01    IF2000     USER EDUCATION                     2
C01    000030     KWAN                               3
C01    000130     QUINTANA                           3
C01    000140     NICHOLLS                           3
```

Problem 3

For all projects (project number) beginning with 'IF', list the following:

First line:

- Text: PROJECT
- Project number
- Employee number responsible for the project
- Estimated starting date
- Estimated ending date

Subsequent line:

- Project number
- Employee number performing the activity
- Activity starting date
- Activity ending date

Display the projects in ascending sequence of their project number and employee number.

Result

	PROJNO	RESPEMP	PRSTDATE	PRENDATE
PROJECT	IF1000	000030	01/01/1982	01/02/1983
	IF1000	000030	01/06/1982	01/01/1983
	IF1000	000130	01/01/1982	01/10/1982
	IF1000	000130	01/10/1982	01/01/1983
	IF1000	000140	01/10/1982	01/01/1983
PROJECT	IF2000	000030	01/01/1982	01/02/1983
	IF2000	000030	01/01/1982	01/01/1983
	IF2000	000140	01/01/1982	01/03/1982
	IF2000	000140	01/03/1982	01/07/1982
	IF2000	000140	01/03/1982	01/07/1982
	IF2000	000140	01/10/1982	01/01/1983

END OF LAB (with hints)

Solution

Problem 1

```
SELECT WORKDEPT, EMPNO, SALARY
   FROM EMPLOYEE
 WHERE WORKDEPT = 'A00'
 UNION ALL
SELECT WORKDEPT, 'SUM', SUM(SALARY)
   FROM EMPLOYEE
 WHERE WORKDEPT = 'A00'
 GROUP BY WORKDEPT
 ORDER BY 1, 2 DESC
```

Problem 2

```
SELECT DEPTNO, 'DEPARTMENT' AS INFO, DEPTNAME, 1
   FROM DEPARTMENT
 WHERE DEPTNO IN ('A00', 'B01', 'C01')
 UNION ALL
SELECT DEPTNO, PROJNO AS INFO, PROJNAME, 2
   FROM PROJECT
 WHERE DEPTNO IN ('A00', 'B01', 'C01')
 UNION ALL
SELECT WORKDEPT AS DEPTNO, EMPNO AS INFO, LASTNAME, 3
   FROM EMPLOYEE
 WHERE WORKDEPT IN ('A00', 'B01', 'C01')
 ORDER BY DEPTNO, 4
```

Problem 3

```
SELECT 'PROJECT', PROJNO, RESPEMP, PRSTDATE, PRENDATE
   FROM PROJECT
 WHERE PROJNO LIKE 'IF%'
 UNION ALL
SELECT ' ', PROJNO, EMPNO, EMSTDATE, EMENDATE
   FROM EMP_ACT
 WHERE PROJNO LIKE 'IF%'
ORDER BY 2, 1 DESC, 3, 4
```

END OF SOLUTION

Exercise 9. Using Subqueries

What This Exercise is About

This exercise provides an opportunity to work with subqueries.

What You Should Be Able to Do

At the end of the lab, you should be able to:

- Code simple subqueries using the '>' and '=' operators
- Code more complex subqueries using the keywords ANY, ALL , and IN

Introduction

See the data model at the start of this exercise guide to get the column names and descriptions for each table.

Required Materials

- Student handout
- SQL Reference

Exercises Instructions

Problem 1

List those employees that have a salary which is $5,000 greater than or equal to the average salary of all employees.

Display department number, employee number, last name, and salary.

Sort the list by the department number and employee number.

Problem 2

List employee number and last name of all employees not assigned to projects. This means that table EMP _ACT does not contain a row with their employee number.

Problem 3

List all employees whose salary is higher than the average salary **of each and every** department in the list 'A01', 'B01', and 'C01'.

Display employee number, last name, and salary.

Problem 4

List all employees whose salary is higher than the average salary **of at least one** of the departments A01, B01, and C01.

Display employee number, last name, and salary.

Problem 5

List project number and duration (in days) of the project with the shortest duration.

Problem 6

List department number, department name, last name, and first name of all employees of departments with only male employees.

END OF LAB

Exercise Instructions With Hints (Expected Results)

Problem 1

List those employees that have a salary which is $5,000 greater than or equal to the average salary of all employees.

Display department number, employee number, last name, and salary.

Sort the list by the department number and employee number.

Result

```
WORKDEPT EMPNO  LASTNAME        SALARY
-------- ------ --------------- --------
A00      000010 HAAS            52750,00
A00      000110 LUCCHESSI       46500,00
B01      000020 THOMPSON        41250,00
C01      000030 KWAN            38250,00
D21      000070 PULASKI         36170,00
E01      000050 GEYER           40175,00
```

Problem 2

List employee number and last name of all employees not assigned to projects. This means that table EMP _ACT does not contain a row with their employee number.

Result

```
EMPNO  LASTNAME
------ ----------
000060 STERN
000120 O'CONNELL
```

Problem 3

List all employees whose salary is higher than the average salary **of each and every** department in the list 'A01', 'B01', and 'C01'.

Display employee number, last name, and salary.

Result

```
EMPNO  LASTNAME        SALARY
------ --------------- ---------
000010 HAAS            52750,00
000110 LUCCHESSI       46500,00
```

Problem 4

List all employees whose salary is higher than the average salary **of at least one** of the departments A01, B01, and C01.

Display employee number, last name, and salary.

Result

```
EMPNO  LASTNAME        SALARY
------ --------------- ---------
000010 HAAS            52750,00
000020 THOMPSON        41250,00
000030 KWAN            38250,00
000050 GEYER           40175,00
000060 STERN           32250,00
000070 PULASKI         36170,00
000110 LUCCHESSI       46500,00
```

Problem 5

List project number and duration (in days) of the project with the shortest duration.

Result

```
PROJNO      DAYS
------ ---------
PL2100      257
```

Problem 6

List department number, department name, last name, and first name of all employees of departments with only male employees.

Result

DEPTNO	DEPTNAME	LASTNAME	FIRSTNME
B01	PLANNING	THOMPSON	MICHAEL
E01	SUPPORT SERVICES	GEYER	JOHN
E21	SOFTWARE SUPPORT	SPENSER	THEODORE
E21	SOFTWARE SUPPORT	MEHTA	RAMLAL
E21	SOFTWARE SUPPORT	LEE	WING
E21	SOFTWARE SUPPORT	GOUNOT	JASON

END OF LAB (with hints)

Solution

Problem 1

```
SELECT WORKDEPT, EMPNO, LASTNAME, SALARY
   FROM EMPLOYEE
 WHERE SALARY >=
                 (SELECT AVG(SALARY + 5000)
                           FROM EMPLOYEE)
 ORDER BY WORKDEPT, EMPNO
```

Problem 2

```
SELECT EMPNO, LASTNAME
   FROM EMPLOYEE
 WHERE EMPNO NOT IN (SELECT EMPNO
                       FROM EMP_ACT)
```

Problem 3

```
SELECT EMPNO, LASTNAME, SALARY
   FROM EMPLOYEE
 WHERE SALARY > ALL(SELECT AVG(SALARY)
                      FROM EMPLOYEE
                    WHERE WORKDEPT IN ('A01','B01','C01')
                          GROUP BY WORKDEPT)
```

Problem 4

```
SELECT EMPNO, LASTNAME, SALARY
   FROM EMPLOYEE
 WHERE SALARY > ANY(SELECT AVG(SALARY)
                      FROM EMPLOYEE
                    WHERE WORKDEPT IN ('A01','B01','C01')
                      GROUP BY WORKDEPT)
```

Problem 5

```
SELECT PROJNO, DAYS(PRENDATE)-DAYS(PRSTDATE) AS DAYS
   FROM PROJECT
 WHERE DAYS(PRENDATE)-DAYS(PRSTDATE) =
       (SELECT MIN(DAYS(PRENDATE)-DAYS(PRSTDATE))
     FROM PROJECT)
```

Problem 6

```
SELECT DEPTNO, DEPTNAME, LASTNAME, FIRSTNME
   FROM DEPARTMENT, EMPLOYEE
 WHERE DEPTNO = WORKDEPT
  AND DEPTNO NOT IN (SELECT WORKDEPT
                            FROM EMPLOYEE
                            WHERE SEX='F')
```

END OF SOLUTION

Exercise 10. Maintaining Data

What This Exercise is About

This exercise provides an opportunity to code SQL statements in order to manipulate data.

What You Should Be Able to Do

At the end of the lab, you should be able to:

- Code UPDATE and DELETE statements with (or without) conditions.
- Code INSERT statements using a VALUE or a SELECT clause.

Introduction

See the data model at the start of this exercise guide to get the column names and descriptions for each table.

Required Materials

- Student handout
- SQL Reference

Exercises Instructions

Problem 1

Create table TESTEMP with the following columns:

- EMPNO
- LASTNAME
- WORKDEPT
- HIREDATE
- SALARY
- BONUS

The data types and null characteristic for these columns should be the same as for the respective columns of the EMPLOYEE table described in the data model.

Problem 2

Mr. SMITH joined the company. For Mr. SMITH enter the following data:

- Employee number: 000111
- Last name: SMITH
- Department number: 'C01'.
- Date hired: 1998-06-25
- Salary: $25,000
- Bonus: $0

Problem 3

Insert data into the TESTEMP table by copying the appropriate columns of the EMPLOYEE table of employees with employee number less than or equal to '000050' into your TESTEMP table.

Problem 4

Our new colleague, Mr. Smith receives a bonus of $500. Make the appropriate data change.

Problem 5

All employees of department 'C01' have done a good job. Therefore they receive a salary increase of $1,000.

Problem 6

After a short time Mr. Smith leaves the company. Perform the data changes.

Problem 7

Insert data into the TESTEMP table by copying the appropriate columns of the EMPLOYEE table of employees with employee number greater than '000050' into your TESTEMP table.

Problem 8

Perform the following changes:

- Theodore Spenser has moved to department 'E01'.

Problem 9

Mrs. Brown joined the company.

For Mrs. Brown enter the following data

- Employee number: 000360
- Last name: BROWN
- Department number: 'D01'
- Date hired: Date when data are entered
- Salary: $45,000
- Bonus: unknown

Problem 10

Clean your test environment.

END OF LAB

Solution

Problem 1

```
CREATE TABLE TESTEMP (
   EMPNO      CHAR(6)        NOT NULL,
   LASTNAME   VARCHAR(15)    NOT NULL,
   WORKDEPT   CHAR(3)                 ,
   HIREDATE   DATE                    ,
   SALARY     DECIMAL(9, 2)           ,
   BONUS      DECIMAL(9, 2)           )
```

Problem 2

```
INSERT INTO TESTEMP
 (EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY, BONUS)
VALUES('000111', 'SMITH', 'C01', '1998-06-25', 25000,
        0)

Rows inserted: 1
```

Problem 3

```
INSERT INTO TESTEMP
 (EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY, BONUS)
SELECT EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY,
       BONUS
   FROM EMPLOYEE
 WHERE EMPNO < = '000050'

Rows inserted: 4
```

Problem 4

```
UPDATE TESTEMP
  SET BONUS = 500
 WHERE EMPNO = '000111'

Rows update: 1
```

Problem 5

```
UPDATE TESTEMP
  SET SALARY = SALARY + 1000
 WHERE WORKDEPT = 'C01'

Rows update: 2
```

Problem 6

```
DELETE FROM TESTEMP
 WHERE EMPNO = '000111'

Rows deleted: 1
```

Problem 7

```
INSERT INTO TESTEMP
            (EMPNO, LASTNAME, WORKDEPT, HIREDATE,
            SALARY, BONUS)
      SELECT EMPNO, LASTNAME, WORKDEPT, HIREDATE,
             SALARY, BONUS
  FROM EMPLOYEE
 WHERE EMPNO > '000050'

Rows inserted: 28
```

Problem 8

```
UPDATE TESTEMP
  SET WORKDEPT = 'E01'
 WHERE EMPNO = '000100'

Rows updated: 1
```

Problem 9

```
INSERT INTO TESTEMP
       (EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY,
        BONUS)
VALUES ('000360', 'BROWN', 'D01', CURRENT_DATE,
         45000, NULL)
```

Problem 10

```
DROP TABLE TESTEMP
```

END OF SOLUTION

CAPITULO IV

4.1 CONCLUSIONES

- El manual del DB2 UDB Family Fundamentals contiene el extracto necesario para el estudio de este sistema de Base de Datos.

- El estudio del DB2 para su respectiva certificación, ayuda al estudiante o profesional a ser más cotizado en la empresa para la cual vaya a trabajar.

- La adquisición de la documentación necesaria para el aprendizaje del DB2 es difícil ya que no existe información necesaria en nuestro medio, y los cursos impartidos son exclusivamente para personas que distribuyen software de la IBM.

- Los exámenes de certificación del DB2 son en Ingles por lo cuál hemos visto la necesidad de elaborar el manual en este idioma, para que de esta manera las personas que se preparen con este documento vayan relacionándose con el idioma y así tengan menos dificultad al rendir el examen.

- DB2 nos brinda un alto rendimiento, facilidad de uso y entendimiento, además soporta diferentes plataformas y nos brinda las seguridades necesarias que puede ser utilizado de acuerdo a las necesidades de las empresas.

4.2 RECOMENDACIONES

- Para sacar la certificación internacional recomendamos que el estudiante use este manual para su preparación.

- Se recomienda que los profesionales o estudiantes se preparen y se certifiquen en DB2, para que de esta manera se les abra nuevos horizontes de trabajo en la vida profesional.

- Adquirir mayores conocimientos en ingles técnico para un mejor entendimiento de la documentación ya que los exámenes son en ingles.

- Que los profesionales estudien y se certifiquen en DB2, por que nos brinda facilidades de uso, soporta varias plataformas y nos da las seguridades necesarias para trabajar.

- Para un buen rendimiento del DB2 Universal Database Versión 7.1 Personal Edition recomendamos Windows 95, 98, NT, 2000 y XP

- Que como requerimientos mínimos sean: Computador Pentium de 233 MHZ, Microsoft Windows 95, resolución de 640x480 Píxeles, 32 MB RAM.

- Que para la Instalación del DB2 UDB V7.1, utilizamos el CD demo sigamos los siguientes pasos:

    - Insertar el CD
    - Esperar que se ejecute el Autorun y escoger la opción "Skip Intro"
    - Seleccionar la opción Try it y ejecutar "DB2 For Windows NT 2000 Personal Edition".
    - Finalmente hacer Clic en Instalar

4.3 VALIDACION

Podemos validar el trabajo desempeñado por los disertantes tomando en cuenta que ambos, Edgar Javier Villacrés Venegas y Edwin Marcelo Gallardo Paredes, obtuvieron la certificación internacional de DB2 UDB Family Fundamentals.

Así como también una evaluación a los estudiantes que recibieron el curso de DB2 para comprobar el nivel de conocimiento de los instructores, calidad del material utilizado en clases, exposiciones claras, confianza a los estudiantes para que expongan dudas, etc.

A continuación adjuntamos los certificados internacionales otorgados por la IBM, la lista de los estudiantes que recibieron el curso, el informe con los resultados expresados en porcentajes de las evaluaciones realizadas.

Todos estos documentos dan constancia de la validación de nuestra disertación.

Lista de Estudiantes que asistieron al curso de DB2 UDB Family Fundamentals.

- Almeida López Cecilia Isabel
- Barragán Benitez Ivan Mauricio
- Bastidas Gallo José David
- Collantes Vaca Santiago Marcos
- Córdova Real Jorge Luis
- Cruz Jácome Elizabeth Diocelina
- García Rodríguez Myriam Jeanette
- León Paredes Gabriel Fernando
- Ortega Sánchez Patricia Paola
- Vayas Ortega Mayra Ivanova
- Vera Altamirano Nancy Geovanna
- Zurita Garzón Adita Teresa

Professional Certification Program from IBM
www.ibm.com/certify

IBM

In recognition of the commitment to achieve professional excellence,
this certifies that

Javier Villacres Venegas

has successfully completed the program requirements as an

IBM Certified Specialist
DB2 UDB V6.1/V7.1 User

Awarded on April 16, 2004
IBM congratulates you on your achievement.

Certified for
IBM DB2
e-business software

Steve Mills
Senior Vice President and Group Executive
IBM Software Group

Janet Perna
General Manager
Data Management Solutions
IBM Software Group

rofessional Certification Program from IBM
ww.ibm.com/certify

IBM

In recognition of the commitment to achieve professional excellence,
this certifies that

Edwin M. Gallardo Paredes

has successfully completed the program requirements as an

IBM Certified Specialist
DB2 UDB V6.1/V7.1 User

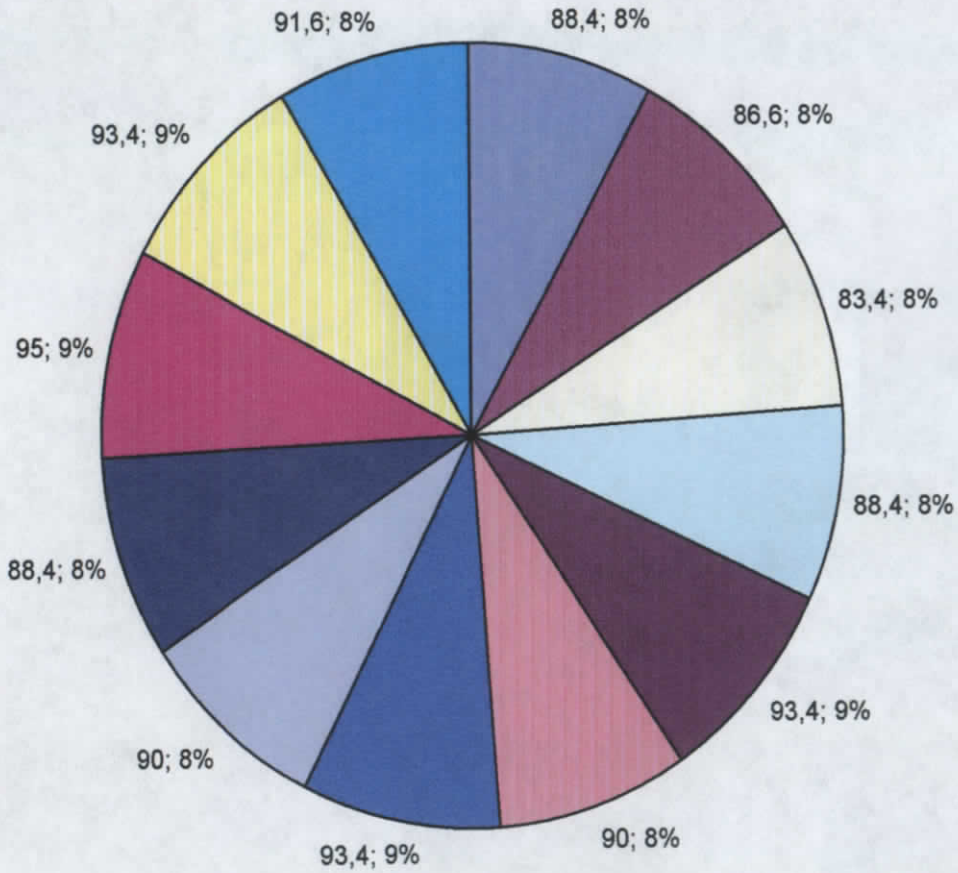
Awarded on March 26, 2004
IBM congratulates you on your achievement.

Certified for
IBM DB2
e-business software

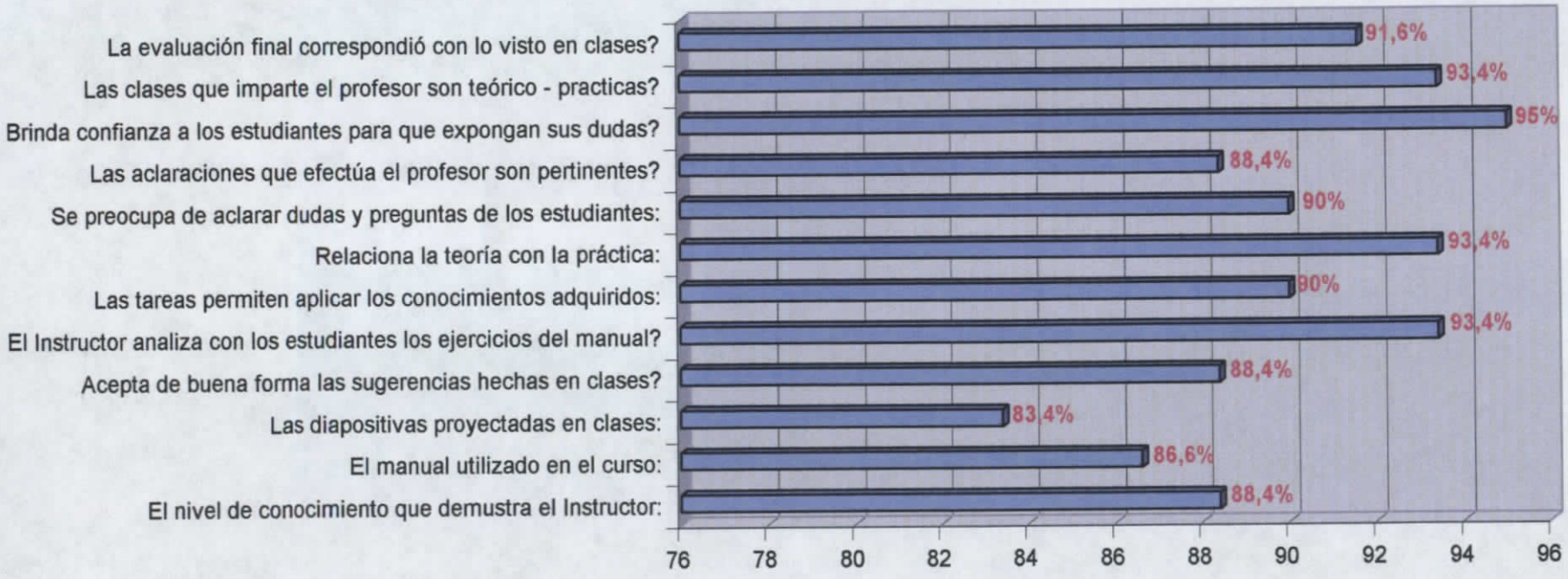
Steve Mills
Senior Vice President and Group Executive
IBM Software Group

Janet Perna
General Manager
Data Management Solutions
IBM Software Group

Evaluación del curso de DB2



Evaluación del curso de DB2



BIBLIOGRAFÍA

PAUL C. ZIKOPOULOS (DB2 Certified Advanced Technical Expert)

"DB2 Universal Database"

LISA CAWLEY (DB2 Certified Advanced Technical Expert)

"DB2 Universal Database"

GEORGE BAKLARZ (Manager of the DB2 Worldwide Pre-sales Support Group)

"DB2 Universal Database"

ROMAN B.MELNYK PhD (Senior member of the DB2 Information Development team)

"DB2 Universal Database"

HANA CURTIS (IBM Certified Solutions Expert)

"DB2 Universal Database"

ROGER E. SANDERS (IBM Certified Advanced Database Administrators)

"DB2 Universal Database"

INTERNET:

http://www.ibm.com

http://www.ibm.com/software/data/db2/library/publications

http://www-3.ibm.com/services/learning/roadmaps

http://www.software.ibm.com

http://www.software.ibm.com/data/highered

http://www.ibm.com/education/certify

http://www-3.ibm.com/software/data/education/cert

http://www.ibm.com/ertify/tests/obj700.shtml

http://www7b.software.ibm.com/dmdd/

http://www7b.software.ibm.com/dmdd/library/tutorials/db2cert/db2cert_v8_tut.html

http://www-1.ibm.com/certify/process/logo_usage.shtml

ANEXOS

EJERCICIOS UNIDAD 4

Problema 1

Listar el número del empleado, apellido, fecha de nacimiento y salario de todos los empleados. Los empleados deben ser ordenados por salario en orden descendente.

Problema 2

Listar el apellido, nombre y el número de departamento de todos los empleados. La lista debe ser ordenada por el número del departamento en forma descendente. Dentro de cada departamento el apellido debe ser ordenado descendentemente.

Problema 3

Listar los diferentes niveles de educación de la compañía en orden descendente. Eliminar los duplicados.

Problema 4

Listar todos los empleados (número de empleado) y los proyectos (número de proyecto). Eliminar los duplicados.

Usar la tabla EMP _ACT.

Problema 5

Listar el apellido, salario y bonos de todos los empleados masculinos.

Problema 6

Listar el apellido, salario y comisión de todos los empleados con un salario mayor que $20,000 y que fueron contratados después de 1979.

Problema 7

Listar el apellido, salario, bono y comisión de todos los empleados con un salario mayor que $22,000 y un bono de $400 o todos los empleados con un bono de $500 y una comisión menor que $1,900. La lista debe ser ordenada por apellidos.

Problema 8

Listar el apellido, salario, bono y comisión de todos los empleados con un salario mayor que $22,000 y un bono de $400 o $500 y una comisión menor que $1,900. La lista debe ser ordenada por apellidos.

Problema 9

Para todos los proyectos 'AD' (número de proyecto que comiencen con 'AD') con actividades de 10, 80 y 180 listar lo siguiente:

- Número de proyecto
- Número de actividad
- Fecha de inicio de la actividad
- Fecha final de la actividad

La lista debe ser ordenada por número de proyecto y número de actividad.

Problema 10

Listar el número del manager y número del departamento de todos los departamentos que tengan asignado un manager.
La lista debe ser ordenada por el número del manager.

Problema 11

Listar el número del empleado, apellido, salario y bono de todos los empleados que tengan un bono entre $800 y $1,000.
Los bonos mas bajos deben aparecer primero.

Problema 12

Listar el número del empleado, apellido, salario y número del departamento de todos los empleados de los departamentos A00 hasta C01.
La lista debe ser ordenada por apellido en forma ascendente.

Problema 13

Listar todos los proyectos (ordenado por número del proyecto) en donde 'SUPPORT' sea parte del nombre.

Problema 14

Listar todos los departamentos que contengan un '1' en la mitad de la cadena de caracteres del número del departamento.

Ordenar la lista por número del departamento.

EJERCICIOS UNIDAD 5

Problema 1

Todos los proyectos que comiencen con 'AD', listar el número de proyecto, nombre del proyecto y número de actividad. Eliminar las filas duplicadas.

Ordenar por número de proyecto y número de actividad.

Problema 2

Cuales empleados fueron asignados al proyecto 'AD3113'? listar el número de empleado, apellido y número de proyecto. Ordenar por número de empleado y número de proyecto. Eliminar los duplicados.

Problema 3

Cuales actividades comenzaron en '1982-10-01'? Para cada una de estas actividades listar el número de empleado que esta ejecutando la actividad, número de proyecto, nombre del proyecto, número de actividad y fecha de inicio de la actividad. Ordenar por número del proyecto, número del empleado y número de actividad.

Problema 4

Listar el número del departamento, número del manager y el nombre de los managers de los departamentos 'A00' y 'B01'

Ordenar por número del departamento.

Problema 5

Listar el número del departamento, apellido, nombre del proyecto y el número de actividad ejecutadas por el empleado del departamento 'A00'.

La lista debe ser ordenada por nombre del proyecto y número de actividad.

Problema 6

Listar el número del departamento, apellido, nombre del proyecto y número de actividad de todos los empleados. Eliminar las filas idénticas.

Ordenar por número del departamento, apellido y número de actividad.

Problema 7

Listar las actividades que sean mayores o iguales a '1982-10-15'.
Para estas actividades listar número de actividad, número de manager responsable para el departamento que esta ejecutando el proyecto, la fecha de inicio de la actividad, el número del proyecto y el apellido de los empleados que están ejecutando la actividad.

Ordenar por número de actividad y la fecha de inicio de la actividad.

Problema 8

Cuales empleados del departamento 'A00' fueron contratados antes que su manager?
Listar el número del departamento, el apellido del manager, el apellido del empleado, la fecha de contratación del manager y del empleado.

Ordenar por el apellido de los empleados.

EJERCICIOS UNIDAD 6

Problema 1

Todos los empleados cuyo salario incrementado un 5 por ciento, sean menores o igual que $20,000, listar lo siguiente:

- Apellido
- Salario
- Salario incrementado en 5 por ciento
- Salario incrementado en 5 por ciento mensualmente

Ordenar por el salario.

Problema 2

Para todos los empleados con un nivel de educación de 18 o 20, el salario debería ser incrementado en $1,200 y el bono reducido a la mitad. Listar el apellido, nivel de educación, el nuevo salario y el nuevo bono para estos empleados.

Listar primero los empleados con un nivel de educación de 20. Para los empleados con el mismo nivel de educación ordenar los salarios.

Problema 3

Los salaries deberían ser reducidos en $1,000 para los empleados que coincidan con los siguientes criterios:

- Los que pertenecen al departamento 'D11'.
- Los que tengan un salario mayor o igual que el 80 por ciento de $20.000.
- Los que tengan un salario menor o igual que 120 por ciento de $20,000.

Listar el número del departamento, apellido, salario y el salario reducido.

Ordenar por salario

Problema 4

Nosotros necesitamos una lista de todos los empleados del departamento 'D11' con un ingreso (Sumar el salario, comisión y el bono) que sea mayor que su salario incrementado en un 10 por ciento. Listar el número del departamento, apellido e ingreso. Ordenar el resultado en base al ingreso en orden descendente.

Problema 5

Listar todos los departamentos que no tengan un manager asignado. Listar el número del departamento y el nombre del departamento.

Problema 6

Listar el número del proyecto y el número mayor del proyecto de todos los proyectos cuyo número de proyecto contenga las letras 'MA' en las posiciones 1 y 2. Si el número mayor del proyecto es desconocido, desplegar el texto 'MAIN PROJECT'.

Problema 7

Listar todos los empleados mayores de 25 años cuando ingresaron a la compañía.

Listar su número de empleado, apellido y edad en años.

Ordenar el resultado por la edad de los empleados cuando ingresaron a la compañía y el número de empleado.

Problema 8

Nosotros necesitamos una lista de todos los proyectos que finalizaron en '1982-12-01'. Desplegar año y mes del inicio de la fecha estimada y el número del proyecto. Ordenar el resultado por el número del proyecto.

Problema 9

Listar el número de proyecto y duración de todos los proyectos que comiencen con 'MA' en semanas. La duración debería ser redondeada y desplegada con una posición decimal.

Problema 10

Para los proyectos cuyo número de proyecto comiencen con 'MA', listar el número del proyecto, la fecha estimada de finalización y la fecha esperada de finalización, si la duración podría exceder lo estimado por el 10 por ciento.

Ordenar la lista por número de proyecto.

Problema 11

Cuantos días hay hasta la finalización de este año? (El resultado esperado fue producido en 1999-04-21 y por consiguiente no coincidirá con tu respuesta)

Tú puedes usar una tabla arbitraria. Especificar un predicado seleccionando una fila.

Para OS/390 tu podrías usar la tabla SYSIBM.SYSDUMMY1 la cual contiene una sola fila.

Problema 12

Listar la inicial del nombre y el apellido de todos los empleados cuyo apellido comience con la letra 'S'. La inicial debe ser separada del apellido por un punto.

Problema 13

Para todos los empleados cuyo número de empleado este entre '000010' y '000130', desplegar los siguientes datos, así como también remplazar algunos valores por un texto:

- Número de empleado
- Los primeros 2 caracteres del número de proyecto reemplazar por el siguiente texto (Usar la columna PROJNO y reemplazar por el alias PROJ):

  AD = ADMIN.
  IF = SUP.
  MA = DEV.
  OP = OPER.
  PL = PLAN.
- Reemplazar el número de la actividad por el siguiente texto (usar la columna ACTNO poner el alias ACT):

  10 = MANAGE
  20 = COST ESTIMATE
  30 = DEFINE SPECS

  Si el número de la actividad no es 10, 20, o 30 desplegar 'OTHER'.
- Fecha de inicio de la actividad
- Fecha de finalización de la actividad

Desplegar las actividades que finalizaron antes de 1989-10-01.

Ordenar el resultado por el número de empleado, la columna PROJ, Fecha inicial de la actividad, la columna ACT y la fecha final de la actividad.

EJERCICIOS UNIDAD 7

Problema 1

Para todos los departamentos, desplegar el número del departamento y la suma de todos los salarios por cada departamento.

Problema 2

Para todos los departamentos, desplegar el número del departamento y el número de empleados

Problema 3

Desplegar los departamentos que tengan más de 3 empleados

Problema 4

Para todos los departamentos con al menos un diseñador, desplegar el número de diseñadores y el número del departamento.

Problema 5

Para cada departamento y cada sexo desplegar el promedio del salario, el promedio del bono, el promedio de la comisión y el número de personas si el grupo sexo contiene al menos 2 personas. El promedio debería ser truncado (no redondeado) con 2 decimales.

Problema 6

Desplegar el promedio del bono y el promedio de la comisión para todos los departamentos con un promedio de bono mayor que $500 y un promedio de comisión mayor que $2,000.

EJERCICIOS UNIDAD 8

Problema 1

Listar el salario de todos los empleados del departamento 'A00'.En la ultima línea del reporte, desplegar la suma de todos los salarios.

Problema 2

Para los departamentos A00,B01 y C01 listar los proyectos que ellos realizados y los empleados que pertenecen al departamento. La salida debería consistir en 3 líneas por cada departamento de la siguiente manera:

Primera línea

- Número del departamento
- Texto: DEPARTMENT
- Nombre del Departamento

Segunda línea (si los datos existen):

- Número del Departamento
- Número del Proyecto
- Nombre del Proyecto

Siguiente línea (Si los datos existen):

- Número del Departamento
- Número del empleado
- Apellido

Problema 3

Para todos los proyectos (número del proyecto) los que comienzan con' IF', Liste lo siguiente:
Primera línea:

- Texto: PROJECT
- Número del Proyecto
- El número del empleado responsable para el proyecto
- La fecha de inicio estimada
- La fecha del fin estimada

Siguiente Línea:

- El número del proyecto
- El número del empleado que realiza la actividad
- La fecha inicial de la actividad
- La fecha final de la actividad

Despliegue los proyectos en la siguiente secuencia número del proyecto y número del empleado en orden ascendente.

EJERCICIOS UNIDAD 9

Problema 1

Liste los empleados que tienen un salario mayor o igual que el promedio del salario de todos los empleados sumado $5,000.
Despliegue número del departamento, el número del empleado, el apellido y el salario.

Ordene la lista por el número del departamento y número del empleado.

Problema 2

Liste el número del empleado y apellido de todos los empleados no asignados a los proyectos. Esto significa que la tabla EMP _ACT no contiene una fila con su número del empleado.

Problema 3

Liste todos los empleados cuyo salario es superior que el promedio del salario de cada uno y cada departamento en la lista' A01,' B01, y' C01.
Despliegue el número del empleado, apellido y salario.

Problema 4

Liste todos los empleados cuyo sueldo es superior que el promedio del salario de por lo menos uno de los departamentos A00, B01 y C01.
Despliegue el número del empleado, apellido y salario.

Problema 5

Liste el número del proyecto y duración (en días) del proyecto con la duración más corta.

Problema 6

Liste número del departamento, el nombre del departamento, apellido, y nombre de todos los empleados de departamentos con empleados masculinos.

EJERCICIOS UNIDAD 10

Problema 1

Cree la tabla TESTEMP con las siguientes columnas:

- EMPNO
- LASTNAME
- WORKDEPT
- HIREDATE
- SALARY
- BONUS

Los tipos de datos y la característica null para estas columnas deben ser igual que las columnas respectivas de la tabla EMPLOYEE descritas en el modelo del datos.

Problema 2

El Sr. SMITH ingresó a la compañía. Para el Sr. SMITH ingresar los siguientes datos:

- El número del empleado: 000111
- El Apellido: SMITH
- El número del Departamento: 'C01 '.
- La fecha de contrató: 1998-06-25
- El salario: $25,000
- Bono: $0

Problema 3

Inserte los datos en la tabla TESTEMP copiando las columnas apropiadas de la tabla EMPLOYEE con el número del empleado menor o igual a' 000050' dentro de la tabla TESTEMP.

Problema 4

Nuestro nuevo colega, Sr. Smith recibe un bono de $500. Haga el cambio del dato apropiado.

Problema 5

Todos los empleados de departamento' C01 ' han hecho un buen trabajo. Por consiguiente ellos reciben un aumento de sueldo de $1,000.

Problema 6

Después de un tiempo corto Sr. Smith deja la compañía. Realice los cambios de datos.

Problema 7

Inserte los datos en la tabla TESTEMP copiando las columnas apropiadas de la tabla EMPLOYEE con el número del empleado mayor que'000050' en la tabla TESTEMP.

Problema 8

Realice los cambios siguientes:
El señor Theodore Spenser ha sido movido al departamento' E01'.

Problema 9

La Señora Brown ingresó a la compañía.
Para la Señora Brown ingresar los siguientes datos

- El número del empleado: 000360
- El Apellido: BROWN
- El número del Departamento:' D01'
- La fecha de contrató: la Fecha cuando los datos son ingresados
- El sueldo: $45,000
- Bonos: desconocido

Problema 10

Eliminar los datos de la tabla TESTEMP

