PNNL-11306

Microsoft SQL Server 6.0® Workbook

E. C. Augustenborg

September 1996

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Richland, Washington 99352

DISCLAIMER

Introduction

This workbook was prepared for introductory training in the use of Microsoft SQL Server Version 6.0.

The examples are all taken from the PUBS database that microsoft distributes for training purposes or from the Microsoft Online Documentation.

Relational Database

- Relationships are implied by data values and do not need to be predefined.
- Based on tables.
- Simpler and more understandable than other models. Easier to maintain.
- Has a special language (SQL) for data definition, retrieval, and update.
- Can define rules that maintain the integrity of the data.

Other types of database models:

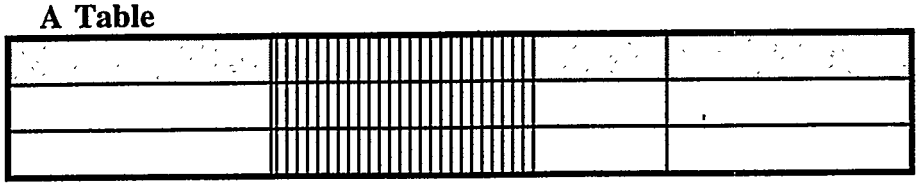
Flat file: Application is integrated with the database record structure

Hierarchical: Each data record is linked to others through the data structure (parents & children)

Network: Random links between records established when the data is defined

Basic Elements of Relational Model

A Table



Attributes (Columns) - describe the domain or entity (Name, Age, Height)

Domain: range of valid entries for a given column.

Instance (Row) : A group of related fields (columns) of information treated as a unit.

Relationships: In a relational database management system, relationships among data are left unstated in the definition of a database. They become explicit when the data is manipulated. However, when you are designing your database you need to give considerable to thought to the relationships between the data you wish to collect.

A table : is a collection of rows that have associated columns. According to the rules of good database design (called normalization rules), each table should describe one type of entity: a person, place, event, or thing. When you want to present or extract information about two or more types of entities, you use the join operation. The join operation is the hallmark of the relational model of database management

Entity-Relationship Diagram: A graphical portrayal of entities and their relationships.

EXERCISE: Do an E-R diagram for a Training Application.

Types of Keys

A key identifies a group of records.

Primary Key: The column(s) whose values uniquely identify a row in a table.

Composite primary key. More than one column is needed to make a unique key.

Foreign Key: A column in one table is the primary key of another tables. Foreign keys are important when defining constraints across tables.

Contrived Key: The key is not a natural part of the data. For example: Item # is generated for each item in an order.

In this version of SQL Server you can enforce **data integrity** by defining the keys when you create the table.

Exercise: Look at some of the tables in the pubs database and describe their keys

Normalization of Tables

Normal Forms: C.F. Codd developed this in order to reduce put the database is a logically consistent form with minimum data redundancy. Each normal form eliminates certain imperfections in the data model. However, normalization does not guarantee that you have the best design ! You may need to denormalize for performance considerations.

1st Normal Form: The value of any attribute in a record must be a single value from the domain of that attribute, not a set of values.

Eg: Multiple values in the Locations field is not allowed

Dept_Name	**Dept_Number**	**Manager**	**Location**
Research	5	333444555	Los Alamos, Richland, Seattle
Support	6	999999999	New York

2nd Normal Form: A relationship is in this form if it is in 1st normal form and all non-key attributes are fully dependent on the primary key.

If the key of the table below is Emp_ID + Project_Number then Employee_name is dependent only a portion of the key.

Emp_ID	Project_Number	Employee Name
123	A55	Mary Smith
456	K44	John Doe

3rd Normal Form: A relationship is in this form if it is in the 2nd normal form and all the nonkey attributes are mutually independent and fully dependent on the primary key.

In the table below Dept_Name is dependent on Dept but the key of the table is Emp_id

Emp_ID	Name	Address	Dept	Dept_Name
123	Mary Smith	Seattle	5	Research
456	John Doe	Los Alamos	6	Support

SQL Preview Using ISQL/W

Titles

Title_id	Title	Pub_id	Price	Ytd_sales
B01032	Cooking with Computers	1389	19.99	4095
MC3021	The Gourmet Microwave	0877	20.95	22246
PC1035	But is it User Friendly	1389	22.95	8780

```
USE        PUBS
GO

SELECT     *
FROM       titles
ORDER BY   title
```

```
Title_id title                                    type       pub_id price    advance         royalty
ytd_sales  notes
pubdate
-------- ---------------------------------------- ---------- ------ -------- --------------- -------
-------- ---------- ---------- ----------------------------------------------------------------------
PC1035  But Is It User Friendly?                  popular_comp 1389  22.95    7,000.00        16
8780    A survey of software for the naive user, focusing on the
'friendliness' of each.                                                    Jun 30 1991 12:00AM
PS1372  Computer Phobic AND Non-Phobic Individuals: Behavior Variations    psychology 0877  21.59
7,000.00        10      375     A must for the specialist,
this book examines the difference between those who hate and fear computers and those who
don't.                                             Oct 21 1991 12:00AM
```

```
SELECT     title_id, title, price
FROM       titles
WHERE      price < 21
```

```
title_id title                                    price
-------- ---------------------------------------- ---------------------
BU1032  The Busy Executive's Database Guide       19.99
BU1111  Cooking with Computers: Surreptitious Balance Sheets  11.95
BU2075  You Can Combat Computer Stress!           2.99
BU7832  Straight Talk About Computers             19.99
MC2222  Silicon Valley Gastronomic Treats         19.99
MC3021  The Gourmet Microwave                     2.99
PC8888  Secrets of Silicon Valley                 20.00
PS2091  Is Anger the Enemy?                       10.95
PS2106  Life Without Fear                         7.00
PS3333  Prolonged Data Deprivation: Four Case Studies  19.99
PS7777  Emotional Security: A New Algorithm       7.99
TC3218  Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean  20.95
TC4203  Fifty Years in Buckingham Palace Kitchens  11.95
TC7777  Sushi, Anyone?                            14.99

(14 row(s) affected)
```

```
SELECT     sum(ytd_sales) Total_sales, avg(price) Avg_price
FROM       titles
```

```
Total_sales Avg_price
----------- ---------------------
```

97446 14.77

(1 row(s) affected)

```
SELECT    title_id, ytd_sales
FROM      titles
WHERE     ytd_sales BETWEEN 4095 AND 9000
```

```
title_id ytd_sales
-------- ----------
BU1032   4095
BU7832   4095
PC1035   8780
PC8888   4095
TC7777   4095
```

(5 row(s) affected)

Publishers

Pub_id	Pub_name	City	State
1389	Algodata Infosystems	Berkeley	CA
0736	New Age Books	Boston	MA
0877	Binnett & Hardley	Washington	DC

```
SELECT    p.pub-name, t.title
FROM      publishers p , titles t
WHERE     p.pub_id   =    t.pub_id
```

```
pub_name                  title
------------------------- ----------------------------------------------------------------------------
Algodata Infosystems       The Busy Executive's Database Guide
Algodata Infosystems      Cooking with Computers: Surreptitious Balance Sheets
New Moon Books            You Can Combat Computer Stress!
Algodata Infosystems      Straight Talk About Computers
Binnet & Hardley          Silicon Valley Gastronomic Treats
Binnet & Hardley          The Gourmet Microwave
Binnet & Hardley          The Psychology of Computer Cooking
                           Algodata Infosystems          But Is It User Friendly?
```

Client Server Paradigm

Client or Front-End

- Gathers data from the user
- Issues a request to the Server
- Presents data to the user
- Many different applications can access the same data (Excel, VB, C)
- Applications may be independent of the Back-end. May start out using Access tables and then move to SQL Server without major changes.
- CPU requirements split between client & server

Server or Back-End

- Processes Requests
- Returns data to the client
- Data integrity can be enforced centrally.

Hardware & Software Requirements

Server

Hardware Platforms:

Alpha AXP microprocessors
Intel 32-bit microprocessors
MIPS microprocessors

O/S: Windows NT version 3.5 or later

Minimum memory: 16 MB.

Minimum Disk Space: 60 MB

Client Utilities

The 32 bit client utilities can be installed on computers running NT Server, NT Workstation or Windows 95.

A Tour of the Tools

SQL Setup

Easy way to install (or reconfigure) SQL Server or Client Utilities. Different setup programs are supplied for each server hardware platform and - for clients - each operating system

SQL Service Manager

Used to start, pause, continue, and stop SQL Server.

ISQL/W

Allows you to enter Transact-SQL statements. Can also analyze queries.

SQL Security Manager

Allows you to manage user accounts for SQL Servers that are using security integration with Windows NT. No need to create special SQL Server login IDs.

SQL Transfer Manager

Provides and easy, graphical way of transferring objects and data from one SQL Server to another.

SQL Performance Monitor

Integrates Windows NT Performance Monitor with SQL Server providing up-to-the-minute activity and performance statistics.

A Tour of the Tools (cont)

SQL Enterprise Manager

- Easy way to administer servers and create database objects.
- Runs on server and 32 bit O/S clients
- Some things you can do:
    - Start & stop the server
    - create and manage devices and databases
    - create and manage tables, views etc
    - create and schedule tasks
    - manage replication
    - execute & analyze queries
    - back up & restore databases
    - generate SQL scripts
    - manage permissions
    - manage server login IDs and database users
    - monitor server activity
    - view the SQL Server error log
- Can use Toolbar or Menus or click Right Mouse button on object
- Click the (+) or (-) sign to expand and collapse lists
- Click on the Display Legend button to find out what each icon means
- First time you start SQL Enterprise Manager you must register the SQL Server you will be managing - give the server name and type of security

ON-LINE BOOKS

Demo how to use

DATABASE

- A collection of tables and other objects
- Each database will contains System Tables and may contain User Tables
- When SQL Server is installed, the setup program installs the following System Databases:
    1. Master (Controls user databases and the operation of SQL Server. Keeps track of user accounts, devices, other databases etc.)
    2. Model (template for new User Databases. Can be customized.)
    3. TempDB (used for temporary storage)
    4. Pubs (sample or training database. Installation is optional.)
    5. MSBD (Used by SQL Executive. Provides a storage area for job scheduling information)

- Users create User Databases. The Model database is the template for User Databases. The model database contains all the system tables required for each user database. All objects in the model database are copied to the new User Database. PUBS is an example of a User Database

- The Data Dictionary is contained in the System Tables. The System Tables in the Master Database are as follows:

Table	Description
syscharsets	Contains one row for each character set and sort order defined for use by SQL Server.
sysconfigures	Contains one row for each user-settable configuration option.
syscurconfigs	Contains an entry for each of the configuration options, as in sysconfigures, but contains the current values and four entries that describe the configuration structure.
sysdatabases	Contains one row for each database on SQL Server.
sysdevices	Contains one row for each disk dump device, diskette dump device, tape dump device, and database device.
syslanguages	Contains one row for each language known to SQL Server. Although U.S. English is omitted, it is always available to SQL Server.
syslocks	Contains information about active locks.
syslogins	Contains one row for each valid SQL Server user account.
sysmessages	Contains one row for each system error or warning that can be returned by SQL Server.
sysprocesses	Contains information about SQL Server processes.
sysremotelogins	Contains one row for each remote user who is allowed to call remote stored procedures
sysservers	Contains one row for each remote SQL Server on which this SQL Server can call remote stored proc.
sysusages	Contains one row for each disk-allocation piece assigned to a database.

- The User Databases contain a subset of the system tables listed above. The system tables in the User Database keeps track of : users, tables, views, indexes, stored procedures, triggers, rules, defaults, user-defined datatypes, permissions

DATABASE OBJECTS

Table: Collection of columns and rows

Index: Ordered set of pointers to data in a table. Allows rapid retrieval of data from a table

View: An alternate way to look at data in one or more tables.

Default: The value that SQL Server inserts into a column if user does not enter a value

Rule: Controls what data can be entered into a table

Stored Procedure: Precompiled collection of SQL statements

Trigger: A special form of a stored procedure that goes into effect automatically when a user modifies data in a table.

To look at the objects in your database:

```
use pubs
go

SELECT    name, type
FROM      sysobjects
```

Object types: C = CHECK constraint; D = Default or DEFAULT constraint;
F= FOREIGN KEY constraint; K=PRIMARY KEY or UNIQUE constraint;
L= Log; P= Stored procedure; R = Rule; RF= Stored procedure for replication;
S = System table; TR = Trigger; U = User table; V = View;
X = Extended stored procedure.

Devices

You can't create a database until you create a device on which to store it. Devices are operating system files used to store databases, transaction logs, and their backups. One device can store many databases, and a database can be stored across more than one device.

There are two types of devices: database devices and dump devices. Database devices store databases and transaction logs and are located on disk files. Dump devices store database and transaction log backups. Dump devices can be stored on disks, diskettes, or tapes

Only the system administrator (SA) can create devices. As administrator, you will need to create database devices to support databases and transaction logs, and dump devices to support backup of those databases and transaction logs.

When you install SQL Server, these devices are created:

- The MASTER database device
- The MSDBDATA and MSDBLOG database devices
- Three dump devices: DISKDUMP, DISKETTEDUMPA, DISKETTEDUMPB.

Because creating and dropping devices makes changes to the system tables in the master database, you should always dump the master database after you add or modify devices.

A database and its transaction log should be kept on separate devices.

The Master device starts out as the default device. You should change this.

Demo:

Create 2 devices - one for the class database and one for its log

CREATE DATABASE

Permission to create a database must be granted by SA.

Defines the :

- Name of the database
- Device(s) on which the database will reside
- Amount of space required on each device
- Amount of space required for the transaction log

Makes a copy of the Model database. If size is not specified, the database will be assigned the size of the Model database

DEMO:

Create database on the new device and a log on the log device

TRANSACTION LOG

SQL Server makes an entry in the syslogs table each time a database page is modified.

Data modifications are always recorded in the log before the change is made to the database itself

Data pages are not written to disk until a checkpoint is issued. Log pages are written to disk when the transaction has completed. Prior to that, data and log pages reside in cache memory.

The transaction log is used during automatic recovery. If the system crashes, SQL Server will use the transaction log to roll forward complete transactions or roll back incomplete transactions. First you should restore the most recent database backup (also called a 'dump') and then all succeeding transaction log backups. Transaction logs must be loaded in the sequence in which they were made.

To restore a database or apply a transaction log

1. From the Server Manager window, select a server.
2. From the Tools menu, choose Backup/Restore.

The transaction log should be kept on a separate device so that

- the database and the log are not competing for the same space. If lots of activity ,the log table can become very large.
- the log can be backed up separately from the database. Frequent backups of the transaction log are necessary.

Truncating a transaction log removes the inactive portion of the log. Truncate the transaction log after you back up an entire database because a database dump does not truncate the log.

Log Size is typically 10-25% of database size but it varies according to the amount of activity and the frequency of transaction log dumps

DEMO

1. Create two dump devices - one for database, one for log
2. Backup CLASS database
3. Do the following insert in the CLASS database : INSERT INTO PUBLISHERS (pub_id, pub_name) VALUES ('0099','TEST RECORD')
4. Write Select stmt to show that record is there
5. Dump the transaction log
6. Make sure no tx are using the CLASS database (click on Current Activity icon)
7. Restore the CLASS database
8. Write Select stmt to show that record is missing
9. Restore the tx log
10. Write Select stmt to show that record is there

SEGMENTS

A segment is a subset of one or more database devices. Particular segments can be allocated to specific tables or indexes.

SQL Server segments can improve SQL Server performance by giving you more control over where you place data on database devices. However, it is better to take advantage of RAID.

Some uses for segments:

- Place a table on one physical device and its nonclustered indexes on a different physical device
- Split a large, heavily-used table acorss databases devices on two separate disk controllers

When a database is created, the following segments are created for the database:

SYSTEM	Stores the system tables
LOGSEGMENT	Stores the transaction log
DEFAULT	Stores all other database objects - unless you create additional segments.

When you add, extend, or delete a segment, you should dump the MASTER database.

When you restore a database with the LOAD DATABASE statement, the segments and devices must be set up in the same way as the database that was dumped.

DATATYPES

Datatypes specify what type of information (characters, numbers, dates) the column can hold and how the data will be stored

System supplied datatypes:

Binary	binary[(n)]	varbinary[(n)]	
Character	char[(n)]	varchar[(n)]	
Date and time	datetime	smalldatetime	
Exact numeric	decimal[(p[, s])]	numeric[(p[, s])]	
Approximate numeric	float[(n)]	real	
Integer	int	smallint	tinyint
Monetary	money	smallmoney	
Special	bit	timestamp	user-defined datatypes
Text and image	text	image	

User defined Datatypes

- Built by the User
- Defined in terms of system supplied datatypes
- If you create user defined datatypes in the Model database they are included in all new databases you create.

EXERCISE

Can add a datatype by typing in the following statement in ISQL/W

EXEC SP_ADDTYPE ZIPCODE, 'CHAR(10)'

or

Use the SQL Executive and click on:
Database Folder
CLASS database
Objects
User Defined Datatypes
Right Mouse button
New UDDT

Example: age smallint

DEFAULTS

Defaults specify the value that SQL Server will insert when a user does not enter a value (in either a NULL or NOT NULL column). For example, in a table with a column called price, if the user does not know the price of an item, you can instruct SQL Server to enter "0" .

A default can be any constant expression

The easiest way to specify defaults is to define DEFAULT constraints when you create a table.

You can also create defaults and then bind them to columns or to user-defined datatypes so that whenever no entry is entered into that column or whenever a column is assigned that user-defined datatype, the default automatically takes effect.

Example

CREATE DEFAULT price_default as 0.00

SP_BINDEFAULT price_default 'titles.advance'

or

Use the SQL Executive and click on:
Database Folder
CLASS database
Objects
Defaults
Right Mouse button
New Default

Type your new default name in the DEFAULTS text box - age_default
Type the default value in the DESCRIPTION box - 25

Press the DATATYPE BINDINGS tab
Bind your default to the 'age' user defined datatype.
Close window

Highlight your new default
Click on Right Mouse button
Generate SQL Script
Preview

RULES

A Rule object contains information that defines a domain of valid values that can be stored in a column or datatype. A rule may be bound to a column or a user-defined datatype.

An alternative method to creating rules is to create table- and column-level CHECK constraints.

Syntax: CREATE RULE [owner.]rule_name AS condition_expression

EXERCISE

Use the SQL Executive and click on:

Database Folder
CLASS database
Objects
Rules
Right Mouse button
New Rule

Type your new rule name in the RULE text box - age_rule
Type the rule in the DESCRIPTION box - @age < 100

Press the DATATYPE BINDINGS tab
Bind your rule to the 'age' user defined datatype.

Other examples:

@title like '[a-z][a-z][0-9][0-9][0-9][0-9]'
@id = 'XYZ' or @id like '_B_'
@city in ('Seattle','New York', 'Moscow')

TABLES

A Table object contains rows of data .

A Column name must be unique with in a table but may be repeated across tables.

For each column you must decide whether or not to allow NULLS.

Use Constraints to protect the integrity of your table

- PRIMARY KEY
- UNIQUE
- FOREIGN KEY
- DEFAULT
- CHECK

IDENTITY[(seed, increment)]

Generates incremental values for new rows based on the seed and increment parameters. The IDENTITY property can be assigned to a tinyint, smallint, int, decimal(p,0), or numeric(p,0) column that does not allow null values. Only one column per table can be defined as an identity column. Defaults and DEFAULT constraints cannot be bound to an identity column, and an identity value cannot be changed.

Types of Tables: System tables, user tables, temporary tables

EXERCISE: Create Table

Use the SQL Executive and click on:
Database Folder
CLASS database
Objects
Tables
Right Mouse button
New Table

1.Create a Course Table with the following columns

Course _ID	smallint	NOT NULL
Title	char(30)	NOT NULL

2.Now click on the Advanced Features icon
Make Course_ID the Primary Key with a Clustered Index
(don't forget to press ADD)
Made Course_ID an Identity column

3.Add a couple of records to the course table

```
insert into course (title)
values ( "SQL Server")

insert into course (title)
values ( "Visual Basic")
```

4. Looks at the new records:
Select * from Course

6. Create a Teachers Table

ID	smallint	NOT NULL
First_name	char(12)	NOT NULL
Last_name	char(20)	NOT NULL
Course_id	smallint	NOT NULL
Annual_salary	money	NULL
Phone_number	char(12)	NULL
Age	age	

7. Click on the Advanced Features icon
Make ID the Primary Key with a Clustered Index (don't forget to press ADD)
Made ID an Identity column
Put a constraint of $10,000 on salary
Make course_id a Foreign Key of the Course table

8. Save the table and call it TEACHERS

9. Add some records to test your constraints

```
insert into teachers (first_name, last_name, course_id, annual_salary)
values ( "Elsa","Augustenborg",1,99999999)
```

```
insert into teachers (first_name, last_name, course_id)
values ( "Fred","Jones",2)

insert into teachers (first_name, last_name, course_id,annual_salary)
values ( "Fred","Smith",3, 9999)
```

10. Look at your new records:

```
Select * from Teachers
```

RETRIEVING DATA

The SELECT statement:

SELECT *list of columns*
FROM *list of tables*
WHERE *search conditions*

Search conditions, or qualifications, in the WHERE clause include:

- Comparison operators (such as =, < >, <, and >)
  WHERE advance * 2 > ytd_sales * price

- Ranges (BETWEEN and NOT BETWEEN)
  WHERE ytd_sales between 4095 and 12000

- Lists (IN, NOT IN)
  WHERE state in ('CA', 'IN', 'MD')

- Pattern matches (LIKE and NOT LIKE)
  WHERE phone not like '415%'

- Unknown values (IS NULL and IS NOT NULL)
  WHERE advance is null

- Combinations of these conditions (AND, OR)

  WHERE advance < $5000 or (ytd_sales > 2000 and ytd_sales < 2500)

EXERCISES : (Refer to your handout of the PUBS tables)

1. Show all information about every author. (Table = AUTHORS)
2. Show all the authors names (first, last) and ids.
3. Show all the authors names and cities who live in the state of California (CA)
4. Show all the authors names and cities who live in CA or UT
5. Show all the authors names who live in a city that begins with "C"
6. List all the cities that have authors. Each city should be listed only once.

7. Show titles and prices books between 5 and 10 dollars. (Table = TITLES)
8. Show all publisher names where the state is not known. (Table = PUBLISHERS)
9. Show all books where the type has a 'p' somewhere in the word and the books cost more than $16.00

10. Select book titles and publisher names for all books where the price is more than $16.00. (Tables = TITLES and PUBLISHERS)

Solutions to 'SELECT' Exercise

1. SELECT * FROM authors

2. SELECT au_fname, au_lname, au_id
   FROM authors

3. SELECT au_fname, au_lname, city,state
   FROM authors
   WHERE state = 'CA'

4. SELECT au_fname, au_lname, city, state
   FROM authors
   WHERE state in ('CA','UT')

5. SELECT au_fname, au_lname, city
   FROM authors
   WHERE city like 'C%'

6. SELECT distinct city
   FROM authors

7. SELECT title, price
   FROM titles
   WHERE price between 5 and 10

8. SELECT pub_name
   FROM publishers
   WHERE state is NULL

9. SELECT title
   FROM titles
   WHERE type like '%p%'
   AND price > 16

10. SELECT pub_name, title
    FROM publishers, titles
    WHERE titles.pub_id = publishers.pub_id

FUNCTIONS

The following functions may be used in SELECT statements. Some of the functions need arguments, others do not. See the Transact SQL manual for exact syntax.

```
Examples:  Select Count(*) from authors
           Select Upper(au_lname) from authors
           Select Substring(au_fname,1,1) + '. ' + au_lname   from authors
           Select Round(price * royalty/100,0) from titles
           Select pubdate, datediff (year, pubdate, "Sep 1, 1996") #_Years from titles
           Select getdate()
            Select      "The price for '" + substring(title,1,20) + "' is $" +
                        ltrim (convert(varchar(10),price))   From titles
           Select base = db_name(),  login=suser_name()
```

Aggregate Functions

Aggregate functions return summary values.

AVG COUNT(*) MIN COUNT MAX SUM

String Functions

String functions perform operations on binary data, character strings, or expressions.

LTRIM SOUNDEX ASCII PATINDEX SPACE CHAR
REPLICATE STR CHARINDEX REVERSE STUFF DIFFERENCE
RIGHT SUBSTRING LOWER RTRIM UPPER

Date Functions

Date functions compute datetime values and their components, dataparts.

DATEADD DATENAME GETDATE DATEDIFF DATEPART

Mathematical Functions

Mathematical functions perform operations on numeric data.

ABS DEGREES RAND ACOS EXP ROUND ASIN FLOORSIGN
ATANLOG SIN ATN2 LOG10 SQRT CEILING PI .TAN
COS POWER COT RADIANS

System Functions

System functions return special information from the database. Some system functions are: HOST_NAME USER_ID DB_NAME USER_NAME HOST_ID

Text and Image Functions

Text and image functions perform operations on text and image data.

PATINDEX TEXTPTR SET TEXTSIZE TEXTVALID

Type-conversion Function

The type-conversion function transforms expressions from one datatype into another.

CONVERT

DATA PRESENTATION

Sorting:

```
SELECT      substring(title,1,40), price
 FROM       titles
ORDER BY    price desc
```

Group By and Having:
Use Group By to divide a table into groups. It is almost always used with aggregate functions. The Having clause is used to restrict results. HAVING is identical to WHERE, except it can include aggregate functions.

Note: If you use ALL with Group By , the query results include all groups produced by the GROUP BY clause, even if some of the groups don't have any rows that meet the search conditions.

Examples:

```
SELECT type, 'Average Price' = AVG(price)
FROM titles
WHERE royalty = 10
GROUP BY all type
ORDER BY type

SELECT    Title = substring(title,1,40),  copies_sold = count(qty)
FROM      sales, titles
WHERE     sales.title_id = titles.title_id
GROUP BY  title
HAVING    count(qty) > 1
ORDER BY  count(qty)  desc
```

Compute BY

COMPUTE BY with row aggregate functions produces reports that summarize values whenever the value in a specified column changes. These summary values appear as additional rows in the query results, unlike the aggregate function results of a GROUP BY clause, which appear as new columns. A COMPUTE BY clause allows you to see both detail and summary rows with one SELECT statement.

Example:

```
SELECT     type, price, advance
FROM       titles
ORDER BY   type
COMPUTE    SUM(price), SUM(advance) BY type
```

DATE MODIFICATION

Adding New Rows:

Insert: Adds new row to a table or a view.

Examples:

```
INSERT titles (title_id, title, type, pub_id, notes, pubdate)
  VALUES ('BU1237', 'Get Going!', 'business', '1389', 'great', '06/18/86')

INSERT stores
  SELECT  '0' + substring(au_id,6,3), au_lname, address, city, state, zip
  FROM    authors
   WHERE state = 'UT'
```

Select Into: If the select into/bulkcopy database option is set, you can use the SELECT INTO clause to create a new permanent table (without using a CREATE TABLE statement). If select into/bulkcopy is not set, SELECT INTO can be used to create only temporary tables.

```
SELECT    pub_id, pub_name
INTO      #newtable
FROM      publishers
```

Note: The above statement creates a temporary table in the tempdb database.

Changing Existing Rows:

Update: Use the UPDATE statement to change single rows, groups of rows, or all rows in a table. Note: a single update statement will never update the same row twice - so use aggregate functions if you have multiple input rows

```
UPDATE publishers
SET city = 'Atlanta', state = 'GA'

UPDATE titles
SET price = price * 2

UPDATE authors
        SET state = 'PC', city = 'Bay City'
                WHERE state = 'CA' AND city = 'Oakland'
```

The following updates the year-to-date sales amount for each title to reflect all sales that occured from 1991 to 1996. Each row in the titles table will get updated once.

```
UPDATE titles
SET ytd_sales = (SELECT SUM(qty)
                  FROM sales
                 WHERE sales.title_id = titles.title_id
                 AND sales.ord_date between '01-01-91' and '12-31-96')
```

```
FROM titles, sales
```

Delete: Removes one or more rows from a table.

Examples: Delete all sales made 5 or more years ago

```
DELETE sales
WHERE datediff(year,ord_date,getdate()) >=3
```

Delete all the books whose authors live in San Francisco
Referential integrity constraint will prevent the following delete

```
DELETE titles
FROM authors, titles, titleauthor
WHERE titles.title_id = titleauthor.title_id
AND authors.au_id = titleauthor.au_id
AND city = 'San Francisco'
```

Transfer Data:

Two SQL Server utilities can be used to transfer data.

- SQL Transfer Manager provides an easy, graphical way to transfer both objects and data from one SQL Server database to another. It can export from a Microsoft-based or non-Microsoft based SQL Server and can import into a Windows NT-based Microsoft SQL Server.

- Bulk copy program (bcp) is a command line utility that copies SQL Server data to or from an operating-system file in a user-specified format.

In addition, SQL Enterprise Manager can be used to generate SQL Scripts, which are the statements that are used to create database objects.

Programming on SQL Server

Transact-SQL : SQL Server version of the SQL database language. Client applications use Transact-SQL to communicate with SQL Server. Transact-SQL allows for creating and manipulating database objects, and for inserting, updating, and selecting data. The Transact-SQL enhancements include data integrity features and stored procedures.

Batches: One or more TRANSACT-SQL statements terminated with a 'GO'. See documentation for rules about which statements cannot be combined in a batch (eg cannot drop an object & recreate it in same batch).

Script: A group of batches.

Example: The following is a script with 2 batches

```
Select ......
Update  .....
Go

Begin Transaction
     Update ........
     Update .......
Commit Transaction
 Go
```

Stored procedures are precompiled static SQL statements. They allow much of an application's processing logic to be run on the Server machine rather than the client. Stored procedures can contain most Transact-SQL statements, including Transact-SQL control-of-flow statements.

Control-of-flow language : Similar to any programming language - allows you to control the execution of SQL statements.

Keyword	Description
BEGIN.... END	Defines a statement block
GOTO	Continues processing at the statement following the label as defined by label.
IF...ELSE	Defines conditional and, optionally, alternate execution when a condition is false.
RETURN	Exits unconditionally
WAITFOR	Sets a delay for statement execution.
WHILE	Repeats statements while a specific condition is true
BREAK	Exits the innermost WHILE loop.
CONTINUE	Restarts a WHILE loop.
RETURN	Exits unconditionally from a query or procedure

Example:

```
Declare @Msg  char(30)

Select  @Msg  = 'No message'
Print   @Msg
```

```
Begin
        If  exists    (Select fname from employee
                        Where fname = 'Paolo')
        Begin
              Delete employee
               Where fname = 'Paolo'
               Select  @Msg  = 'Paolo is no more !!!'
        End
        Else
              Select  @Msg  =   'Sorry - Could not find Paolo'
End
Print @Msg
go
```

Feature	Description
CASE	Allows an expression to have conditional return values
Comments	Inserts a comment anywhere in an SQL statement. Two commenting styles are supported: Transact-SQL style (/* and */) and ANSI-standard comment style (--).
DECLARE	Declares local variables as well as cursors
PRINT	Prints a user-defined message on the user's screen
RAISERROR	Returns a sysmessages entry or a dynamically built message with user-specified severity and state. RAISERROR also sets a system flag (in the global variable @@ERROR) to record that an error condition has occurred

Stored Procedures

Advantages: Precompiled - speed, syntax checked. Can be shared by many applications. Can serve as security mechanism.

Example:

First create the stored procedure

```
If Exists (Select *
          From sysobjects
          Where id = object_id('dbo.usp_author_info') )
  Begin
       drop procedure dbo.usp_author_info
  End
Go

CREATE PROCEDURE usp_author_info
AS
  SELECT ltrim(au_lname) + ', ' +substring(au_fname,1,12) Name, title,
         pub_name
  FROM authors a, titles t, publishers p, titleauthor ta
  WHERE a.au_id = ta.au_id
  AND t.title_id = ta.title_id
  AND t.pub_id = p.pub_id

Go

Grant execute on usp_author_info to public
Go
```

Then execute the stored procedure

```
EXECUTE usp_author_info
```

You may view your stored procedure in SQL Executive and make changes to it there.

Using Input and Output parameters with stored Procedures

```
CREATE PROCEDURE titles_sum
        @title varchar(40) = '%',
       @sum money OUTPUT
AS
SELECT 'Title Name' = title
        FROM titles
                WHERE title LIKE @title
SELECT @sum = SUM(price)
        FROM titles
                WHERE title LIKE @title

DECLARE @totalcost money
EXECUTE titles_sum 'The%', @totalcost OUTPUT
IF @totalcost < 200
BEGIN
        PRINT ' '
        PRINT 'All of these titles can be purchased for less than $200.'
END
ELSE
        SELECT 'The total cost of these titles is $' + rtrim(convert(varchar(20),
                @totalcost))
go
```

System Stored Procedures : make it easy to retrieve information from the system tables, administer databases, and perform other tasks that involve updating system tables. These stored procedures are less necessary with the advent of SQL Executive.

Example: sp_helpdb

With Recompile

The WITH RECOMPILE clause is helpful when the parameters you supply to the procedure won't be typical.

```
CREATE PROCEDURE titles_by_author
        @lname_pattern  varchar(30) = '%'
WITH RECOMPILE
```

Triggers

A Trigger is a special type of stored procedure that is automatically executed when data in a specific table is inserted, updated, or deleted. In the past triggers were used to enforce referential integrity. Can also use them to enforce business rules. For example - no more than 7 students may be enrolled in a particular class.

Each table can have up to 3 triggers for INSERT, UPDATE, DELETE.

A trigger that changes data in another table will invoke triggers on that table. However, a trigger will not call itself.

Triggers cannot return query results to the user.

Certain SQL statements such as DROP are not allowed in Triggers.

There are two special tables used by triggers:

INSERTED: When a row is inserted into a table it is also recorded in a special inserted table.

DELETED: When rows are deleted from a table, they are placed in a special deleted table.

Note: An Update moves the original row in the DELETED table and puts the new row in the INSERTED table. These tables are stored in memory not on disk - so access is very fast. They are structurally like the table on which the trigger is defined.

The trigger is the last thing executed. If anything goes wrong the entire transaction is rolled back. You cannot create a trigger on a view or a temporary table. A 'Truncate table' statement does not active the triggers on the table being truncated

To Create a New Trigger

In SQL Enterprise Manager,
Select a table from the Tables drop-down list
Click right Mouse Button
Choose Triggers from drop-down list.

Trigger Example:

The following trigger does not allow an update on a particular column

```
CREATE TRIGGER u_tr_employee
ON employee
FOR UPDATE
 AS
        IF Update(hire_date)
        Begin
            Print "You may not update the employee hire date"
            Rollback transaction
        End
```

Now test your new trigger:

```
UPDATE employee
        SET hire_date = getdate()
                WHERE emp_id = 'MPA42628M'
```

If several records are inserted at one time the following trigger will ensure that the good records get inserted while the records that failed get rolled back.

```
CREATE TRIGGER i_tr_onlygood
ON sales
FOR INSERT AS
        IF
        (SELECT COUNT(*) FROM titles, inserted
        WHERE titles.title_id = inserted.title_id) <> @@rowcount
        BEGIN
                DELETE sales FROM sales, inserted
                WHERE sales.title_id = inserted.title_id AND
                        inserted.title_id NOT IN
                                (SELECT title_id
                                 FROM titles)
                PRINT 'Only sales records with matching title_ids added.'
        END
```

SECURITY CONCEPTS

Protects a server and the data stored on that server

3 Types of Security

Integrated: uses Windows NT authentication mechanisms to validate logins. Only trusted connections are allowed - not all network protocols have trusted connections. Can login to SQL Server withough supplying an ID and password. After login all other security checks are handled as normal.

Standard: uses SQL Server's own login validation process for all connections. User must provide login id and password which are validated against entries in the syslogins table in the Master database.

Mixed: Allows login requests to be validated using either integrated or standard security.

3 Types of Special Users

SA: system administrator. Is responsible for the server as a whole, not a particular database or application. Duties include: installation, configuration, monitoring, creating devices & databases, granting permissions, backup and restore. SQL Server does no permission checking for the SA. SA can do everything and does not need to have permissions granted.

DBO: database owner. Is the user who creates a database. Only one login ID can be DBO although other login IDs can be aliased to DBO. Can do everything in the database that he/she created and can grant permissions to others in that database.

Database Object Owner: the user who creates the database object such as table, index, view, trigger. The owner is automatically granted all permissions on the database object. The database object owner can grant permissions to other users to use that object. Ownership cannot be transferred.

CREATING AND MANAGING INDEXES

Definition: An Index object is used to speed up access to a table. There are two types of indexes:

- a clustered index, where data is stored in the same order as the index. Can have only one per table. Retrieval is faster (especially if you are searching for a range of records) than a nonclustered index but modifications may be slower.
- a nonclustered index, where data is unordered and stored separately from the index.

When retrieving data, the optimizer considers the amount of work involved if the index is used vs the amount of work involved if the index is not used. The decision is based on statistics. Keep statistics up-to-date. The optimizer will do a table scan rather than use an index when it expects to return a large % of the rows. For composite indexes, if you do not specify all the columns, you must specify columns starting from the left - if the index is to be used.

SQL Server automatically creates an index for the PRIMARY KEY and UNIQUE constraints. If you want other indexes, you must create them.

To create an index

In SQL Enterprise Manager
select the appropriate table
click the right mouse button
choose Indexes.

Other Security Concepts

Username or User: A database username is a name known to the database and assigned to a login id for the purpose of allowing a user to have access to the database. Permissions are granted to the username or the a group to which the user may belong.

Group: A collection of database users. It is easier to manage permissions at a group level than at a user level. Each user belongs to the 'Public' group by default. Each user may belong to one other group.

Alias: Is a database username that is shared by several login IDs. A common use for alias is to allow several users to assume the role of DBO. For example:

- assign the new username 'vp' to a login ID
- grant permissions in the database to 'vp'
- assign several other login IDs to the alias 'vp'

Permissions:

- ***Object Permissions*** - actions you are allowed to perform on an object (Select, Update, Insert, Delete, Reference, Execute). These permissions are granted and revoked by the object owner.

- ***Statement Permissions*** - SQL statements that you are allowed to execute. (Create database, create table, dump database, create view...) These permissions are granted or revoked by the SA or DBO and apply to only 1 database.

Views & Stored Procedures: If granted access to a view or stored procedure, you have access to all the objects that the view or stored procedure reference - even if you have not been granted direct access to these objects.

How to Set Up Integrated Security

1. Use SQL Enterprise Manager to set the Server's security options
   - Open a server group & select the server
   - Choose Configurations
   - Choose the Security Options Tab
   - Review and change the options as necessary
   - When finished choose the OK button
   -

2. Create Windows NT groups and users - using the Windows NT User Manager.
   - For example: create a local group names SQLUsers that has user-level priveleges, and another named SQLAdmins that has system administrator priveleges. Then add individual users to these groups.
   - Avoid placing NT users in more than one NT group that can access SQL Server, because SQL Server does not allow overlapping group membership within databases.

3. Authorize selected Windows NT groups and users to access SQL Server - using the SQL Security Manager to map the NT groups and users to SQL Server login Ids (or the default login ID)
   - Start SQL Security Manager and log into a SQL Server
   - Choose the User Privilege button
   - From the Security menu, choose Grant New
   - In the Grant privilege box, select the group that will have access to SQL Server
   - To add users in the group to a database and make that database their default database - select Add Users to Database box and then select the database from the list.
   - Choose the Grant button.

4. Set object permissions for each databases - using SQL Enterprise Manager.
   - Select a server, open its Databases folder and select a databse
   - From the Object menu, select Permissions
   - Select the By User tabl in the Permissions box
   - From the User/Group list, select a user or group
   - To limit the amount of information displayed, select or clear the Object Filters options.
   - To grant to the selected user or group all permissions for all displayed objects, choose Grant All
   - To grant or revoke specific permission for specific objects, click on the appropriate check boxes in the object list.
   - Choose Set.

5. Set statement permissions for each database - using SQL Enterprise Manager.
    - Open the Edit Database dialog box for the database that will have its statement permissions administered.
    - Choose the Permissions tab
    - To assign a permission, select that box
    - Choose OK