



**SQL Server 2000 Handbook**  
2<sup>nd</sup> Edition

Author: Stamati Crook  
stamati@redware.com  
Date: 18 October 2001  
Version: 4.0

© REDWARE 1996, 2001.

## Shareware Licence

### Copyright © REDWARE 1996, 2001.

8 September 2001 - Version 4.0

All rights reserved. This book is **shareware** and may be downloaded and stored on a single computer for 30 days for the purposes of evaluation only. Registration is required by making the appropriate payment at the **redware** website. The book is copyright and no part shall be reproduced, stored in a retrieval system, or transferred by any means: electronic, mechanical, photocopying, recording, or otherwise without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this handbook, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein. For information, please contact:

**redware research ltd**, 104 Tamworth Road, Hove BN3 5FH, England.

<http://www.redware.com>

## Acknowledgements

### Second Edition September 2001

Thank you this time to Victor, Phong and especially James for helping build really big database systems during our roller coaster ride at First Telecom. Thanks also to the job market for letting me take a break and update this book.

### First Edition December 1996

Thank you to the technical team at F1 Computing Systems past and present for eight years of implementing FoxPro projects. All of my FoxPro experience has resulted from working with Ian, Phong, James, Danny and David on various projects during my eight years at F1. They still have the best training courses and FoxPro team in the UK.

Thank you especially to James Thornton at F1 Computing who put me straight on a few things regarding SQL Server. Any errors remaining in this book are, of course, down to me and I apologise in advance for them. Please email me with your comments, good and bad.

# 1. Contents

---

<b>1. CONTENTS.....</b>	<b>3</b>
<b>2. SQL SERVER OVERVIEW .....</b>	<b>6</b>
RELATIONAL DATABASE TERMINOLOGY .....	6
SQL SERVER HISTORY.....	6
<i>Sybase</i> .....	6
<i>SQL Server 4.2</i> .....	6
<i>SQL Server 6.0</i> .....	7
<i>SQL Server 6.5</i> .....	7
<i>SQL Server 7.0</i> .....	7
<i>SQL Server 2000</i> .....	7
SQL SERVER FEATURES .....	8
<i>Transactions</i> .....	8
<i>Data Dictionary</i> .....	8
<i>Constraints</i> .....	8
<i>Structured Query Language</i> .....	8
<i>Enterprise Networking</i> .....	9
<i>Administration</i> .....	9
<i>Connectivity</i> .....	10
<i>Views</i> .....	10
<i>Triggers</i> .....	10
<i>Stored Procedures</i> .....	10
<i>Replication</i> .....	11
<i>User Defined Functions</i> .....	11
<i>XML</i> .....	11
<i>HTML</i> .....	11
<b>3. SQL TOOLS .....</b>	<b>12</b>
SERVICE MANAGER.....	12
ENTERPRISE MANAGER .....	12
<i>Register the Server</i> .....	13
SQL QUERY ANALYSER.....	14
OSQL .....	15
<b>4. SQL SYNTAX.....</b>	<b>17</b>
PUBS .....	17
SELECT STATEMENT .....	17
<i>Field List</i> .....	18
<i>WHERE Clause</i> .....	18
<i>Wild Cards</i> .....	19
<i>FROM Clause</i> .....	19
<i>ORDER BY</i> .....	20
<i>Natural Join</i> .....	20
<i>GROUP BY Clause</i> .....	21
<i>HAVING Clause</i> .....	21
<i>DISTINCT</i> .....	21
<i>Inner (Natural) Join</i> .....	22
<i>Outer Join</i> .....	22
<i>Sub Queries</i> .....	22
<i>UNION</i> .....	23

FOR XML mode [ , XMLDATA ] [ , ELEMENTS ] [ , BINARY BASE64 ] .....	23
SELECT .. INTO .....	24
INSERT STATEMENT .....	24
UPDATE STATEMENT .....	24
DELETE STATEMENT .....	25
<b>5. DATABASE DEFINITION .....</b>	<b>26</b>
ENTERPRISE MANAGER .....	26
CREATE A DATABASE .....	27
CREATE A TABLE .....	29
Data Types .....	30
Nulls and Defaults .....	31
Table Ownership .....	31
FIELD PROPERTIES .....	32
Null Values .....	33
DEFAULT CONSTRAINTS .....	34
CHECK CONSTRAINTS .....	34
CREATE A PRIMARY KEY .....	36
Identity Columns .....	36
Unique Identifiers .....	37
PRIMARY KEY CONSTRAINT .....	38
FOREIGN KEYS AND REFERENTIAL INTEGRITY .....	39
USER DEFINED DATA TYPES .....	41
DEFAULTS AND RULES .....	41
Defaults .....	42
Rules .....	42
<b>6. INDEXES .....</b>	<b>43</b>
UNIQUE INDEX CONSTRAINT .....	43
CLUSTERED INDEX .....	43
<b>7. VIEWS .....</b>	<b>45</b>
INDEXED VIEWS .....	46
CHECK OPTION .....	46
PARTITIONED VIEWS .....	46
OPENROWSET .....	47
LINKED SERVERS .....	47
TEMPORARY TABLES .....	48
<b>8. STORED PROCEDURES .....</b>	<b>49</b>
EXECUTING A STORED PROCEDURE .....	50
PASSING PARAMETERS .....	52
RETURNING A VALUE .....	53
OUTPUT PARAMETERS .....	53
PROGRAM STRUCTURES .....	54
LOCAL VARIABLES .....	55
SYSTEM VARIABLES .....	56
SCALAR FUNCTIONS .....	57
CASE EXPRESSION .....	57
CURSORS .....	58
SYSTEM PROCEDURES .....	59
EXTENDED PROCEDURES .....	60
EXTENDED MAIL PROCEDURES .....	61

ERROR HANDLING.....	61
TRANSACTIONS .....	63
DISTRIBUTED TRANSACTIONS .....	64
<b>9. TRIGGERS.....</b>	<b>66</b>
TRIGGER PROGRAM STRUCTURE.....	66
FIELD LEVEL VALIDATION.....	68
RECORD LEVEL VALIDATION .....	69
CHECKING VALUES AGAINST ANOTHER TABLE.....	69
PREVENTING CHANGES TO A FIELD.....	69
REFERENTIAL INTEGRITY CHECKS .....	70
<i>Checking a Foreign Key</i> .....	70
<i>Ensuring Unique Candidate Keys</i> .....	71
<i>Checking Referential Integrity on Delete</i> .....	71
CASCADING DELETE .....	71
UPDATING ANOTHER TABLE.....	72
<b>10. SQL SERVER OPTIMISATION.....</b>	<b>74</b>
QUERY OPTIMISATION .....	74
<i>Update Statistics</i> .....	74
<i>Index Design</i> .....	74
<i>Ordering</i> .....	76
<i>Showplan</i> .....	76
<i>SQL Trace</i> .....	77
<i>Optimiser Hints</i> .....	78
CLUSTERED INDEXES .....	78
INDEX TUNING WIZARD.....	79
STORED PROCEDURE RECOMPILATION.....	79
DEFERRED UPDATES .....	80
LOCKING ISSUES .....	80
<b>11. CONFIGURATION .....</b>	<b>82</b>
SERVER CONFIGURATION.....	82
<i>Memory</i> .....	82
<i>Lock Escalation Percentage</i> .....	83
<i>Network Packet Size</i> .....	83
<i>Open Databases</i> .....	83
<i>User Connections</i> .....	83
DATABASE CONFIGURATION.....	84
<i>Size of tempdb</i> .....	84
<i>Truncate Log on Checkpoint</i> .....	84
<i>Deleting a Database</i> .....	84
BACKUPS.....	84
<b>12. SECURITY .....</b>	<b>85</b>
SERVER LOGINS .....	85
DATABASE USERS .....	86
<i>Permissions</i> .....	87
<b>13. INDEX .....</b>	<b>90</b>

## 2. SQL Server Overview

---

### Relational Database Terminology

**Relational** database theory was first defined by Edgar Codd on the principle that relationships between database tables could be defined by the programmer rather than implicitly in the database definition. This improved on the flexibility of the hierarchical database and allowed the programmer to join any two tables together on any common field as required at the application level.

The relational model defines **tables** as a collection of **fields** (domains) which contain values stored as **records** (tuples) in the table. Each record must have a **primary key** which uniquely identifies the occurrence of the record within the table. Fields are defined as numeric, character, date and so forth and may or may not contain values. A relational database can distinguish between a blank or zero and an empty or **null** value.

The programmer may define a **join** between two tables on any common field. This is usually determined by the database designer who includes **foreign key** values in the child datafile that contain primary key values of the parent datafile to allow corresponding records to match up. The programmer may however join tables on any field or fields of the same datatype in both tables to create a **many-to-one** or **one-to-one** relationship. Note that **many-to-many** relationships may not be implemented in a relational database and are implemented with a virtual link table containing foreign key relationships to each of the parent tables.

A Database Management System (DBMS) usually has a **database definition language (DDL)** which allows for the field types and tables to be defined and a **data manipulation language (DML)** which allows for the retrieval and update of data. The manipulation language often comes in several formats allowing access to the database from a variety of programming languages.

Codd went on to define a combined database definition and data manipulation language called **Structured Query Language** or **SQL** (pronounced Sequel). This was implemented in IBM's first relational database product and has now become the standard for most relational database systems. Many older hierarchical and network database management systems also allow data manipulation by interpreting SQL syntax to perform operations on data stored in more traditional logical database formats.

### SQL Server History

#### Sybase

Sybase come into the fray over ten years ago as a pure implementation of an RDBMS taking into account many of the technical refinements of the first generation RDBMS. The implementation is functionally equivalent and an effective competitor to relational database products from Oracle, Ingres and IBM DB2. Sybase runs on many large UNIX computers and is compatible to a large extent with SQL Server.

#### SQL Server 4.2

Microsoft licensed Sybase technology for use on their operating systems and SQL Server is an implementation of Sybase 4.2 on the OS/2 and Windows NT platforms. The NT version offers the technical advantages of the Sybase implementation coupled with a visual administration tool and very cost effective transaction rates.

**SQL Server 6.0 Splash Screen**



**SQL Server 6.0**

SQL Server 6.0 is a rewrite of the original SQL Server product that takes advantage of the Windows NT Operating System and allows remote management of a collection of enterprise wide servers. Microsoft are following an independent path from Sybase and have incorporated advanced features such as Replication and support for multi-processor hardware in this version.

**SQL Server 6.5**

SQL Server 6.5 contains several performance improvements particularly in areas where many users are accessing the same portion of a table for updates. This improves various contention scenarios when many users are attempting to add records and compete to add sequentially to a clustered index for example.

Replication is also much improved and can now replicate with other ODBC data sources as well as interface with Oracle or other more complicated corporate situations.

**SQL Server 7.0**

SQL Server 7.0 re-engineered the product to use native Windows NT files for a more logical integration with backup systems. Many of the configuration parameters became 'self-tuning' to avoid the need for a DBA on smaller systems.

**SQL Server 2000**

More improvements for the DBA including an index tuning wizard which suggests potential indexes to be placed on tables. Introduction of user defined functions and partitioned views and functionality to support XML.

## SQL Server Features

SQL Server is a fully fledged relational database server that runs on all versions of Microsoft Windows. The server software is licensed from Sybase and there is a high degree of compatibility with large scale Sybase servers.

### Transactions

Many DBMS allow for the concept of a transaction which is a programmer defined unit of work. The programmer defines the beginning and the end of the transaction and any changes made to the data in the database are logged in a transaction log until the programmer completes the transaction with a **Commit** command. The database will then write all of the transactions into the database. If there are any problems in completing the transaction, for example a record locking deadlock occurs with another user, then the DBMS will **Rollback** the database as if the transaction never happened.

The transaction log may also help with database recovery in case of a hardware failure in that the database can be rolled forward using the transaction log from a previously saved state until the last fully completed database transaction.

Correct use of programmer defined transactions allows for the data stored in the database to be correct at all times even if a hardware failure interrupts the program flow.

### Data Dictionary

Usually an RDBMS will support a **data dictionary**. This is a set of tables which are stored in each user database and are referred to as **system tables**. SQL Server maintains several system tables each containing information about different parts of a database. For example a system table, called SYSINDEXES, exists in each user database which contains information about all the indexes set-up on tables across the users database.

These system tables can be queried and viewed like any other table but are usually hidden from the user to avoid confusion and can also be accessed using system stored procedures.

### Constraints

Constraints may often be defined in a Database to allow data to be checked by the database software before it is added or modified in the database. This has the advantage of ensuring that data is always valid as a program cannot pass in data that breaks a constraint and also allows these checks to be implemented once in the database software rather than in each application that updates the database.

### Structured Query Language

The **structured query language (SQL)** used in SQL Server is very similar to the ANSI SQL standard. Following are a few examples of SQL commands, more detailed explanations of the commands available can be found in the Transact-SQL Reference manual supplied with SQL Server.

Creation of a table:

```
CREATE TABLE contact
(contact_id cid,
name varchar(30),
address varchar(60),
telephone_number varchar(20),
rating tinyint)
```

Insertion of a record into a table:

```
INSERT INTO TABLE contact
  (contact_id,
   name,
   address,
   telephone_number,
   rating)
VALUES
  ('00003215',
   'John Brown',
   '67 North Street, Guildford, Surrey',
   '0327-7384629',
   8)
```

Updating of a record:

```
UPDATE contact
  SET telephone_number = '01327-7384629',
      rating = 9
  WHERE contact.contact_id = '00003215'
```

Deletion of a record:

```
DELETE contact
  FROM contact
  WHERE contact.contact_id = '00003215'
```

In addition to the standard SQL functions, SQL Server supports extensions to the traditional syntax to allow the implementation of outer joins and other enhancements to the language.

SQL Server also implements a programming version of the SQL language known as Transact SQL which is used in the definition of program scripts for triggers and stored procedures.

### **Enterprise Networking**

SQL Server is part of the BackOffice suite of programs designed to run on Microsoft NT Advanced Server. The technology is suitable for Enterprise Networking where many NT Servers are situated throughout an organisation connected together in a Wide Area Network.

SQL Server may be installed on some or all of the servers to provide departmental databases. Programs may access more than one database if required. In addition, a SQL user may be configured as a remote user on another server so that the two servers communicate by automatically logging the user onto the second, remote, server to allow access to data.

Further facilities such as security that is integrated with network security, replication of data between servers, remote administration of servers from a workstation, and integration with electronic mail make SQL Server a good choice for a multi-server networked environment.

### **Administration**

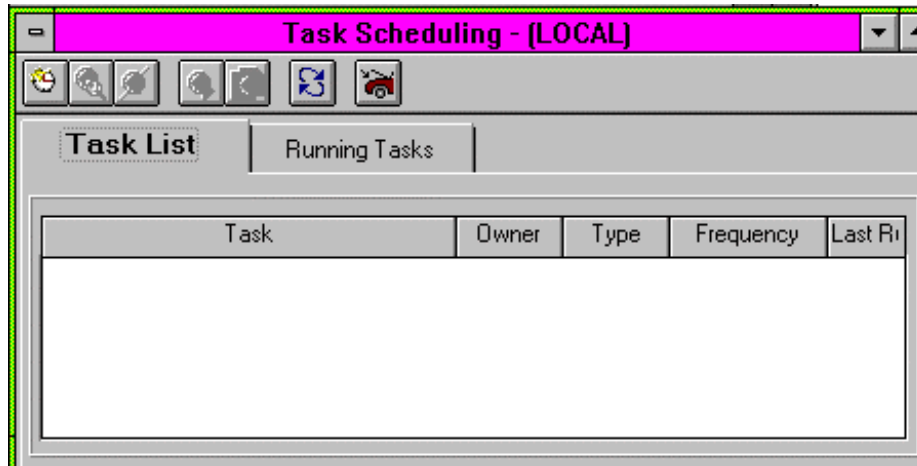
Administration of the server is performed through the SQL Enterprise Manager which allows for the management of any server on the LAN or WAN using client software running on Windows 95 or Windows NT.

Microsoft have implemented software components which can connect to SQL Server administration functionality for programmatic control of complex administration and system management.

SQL Server has a task management program that schedules activity at regular intervals. This activity includes the implementation of replication triggers which replicate data between servers.

The current Tasks can be viewed with the Tools-Task Scheduling... menu option in the SQL Enterprise Manager.

### **Task Scheduling Window**



User security may be automatically inherited from the Network Configuration and the specification of physical devices for the database is fairly straightforward. The database can take advantage of sophisticated operating system features such as RAID fault tolerant disks to supplement the security features of mirrored transaction logs.

### **Connectivity**

Microsoft provide the latest ODBC and OleDb drivers for SQL Server to provide some of the highest connection speeds available from a variety of programming environments. ADO ActiveX data object provide a convenient way to manage OleDb data sources and a well integrated with a SQL Server environment.

Gateways exist to transparently connect a request for SQL Server data through to a Mainframe database.

### **Views**

Views on data may be easily defined to allow local or global corporate database schemas to be defined and yet allow for the underlying local structure to be changed if required without affecting existing programs.

### **Triggers**

Triggers allow programs to be executed on the server whenever data is updated to prevent updates or to perform processing,

### **Stored Procedures**

Stored procedures are programs that run on the server using an enhanced form of SQL called Transact-SQL. This includes program control functionality and the facility to call external programs residing on the server such as electronic mail.

There are considerable benefits in getting the server to perform tasks rather than calling a workstation process. This is particularly relevant in high transaction systems which interface with

other components of the computing infrastructure as network traffic is not a bottle-neck when the processing is performed solely on the server,

### **Replication**

Data is published on one server and other servers are defined as subscribers to that data. The SQL Executive copies data regularly during the process of database synchronisation. Replicated data is not modifiable on the subscription databases.

### **User Defined Functions**

User defined function (new in SQL 2000) allow Transact-SQL to be used to create a program that returns a single value or one that returns a cursor. The former allows re-usable functions to be used in program implementation and the latter provides a programmatic alternative to defining a View.

### **XML**

XML has considerable support in SQL 2000 and allows fully formed XML files to be returned directly from a server stored procedure without the need of any further middleware.

### **HTML**

SQL Server has been integrated with Internet Information Server to serve directly to an http: request. This can provide powerful functionality when used together with stored procedures return XML in combination with an XLST formatting file.

## 3. SQL Tools

---

### Service Manager

SQL Server runs as a service on Windows NT Advanced Server and is usually configured to autostart. The SQL Service Manager can be used to start SQL Server running and must be run on the server itself. Simply bring up the window and press the buttons to start and stop the server.

#### SQL Service Manager

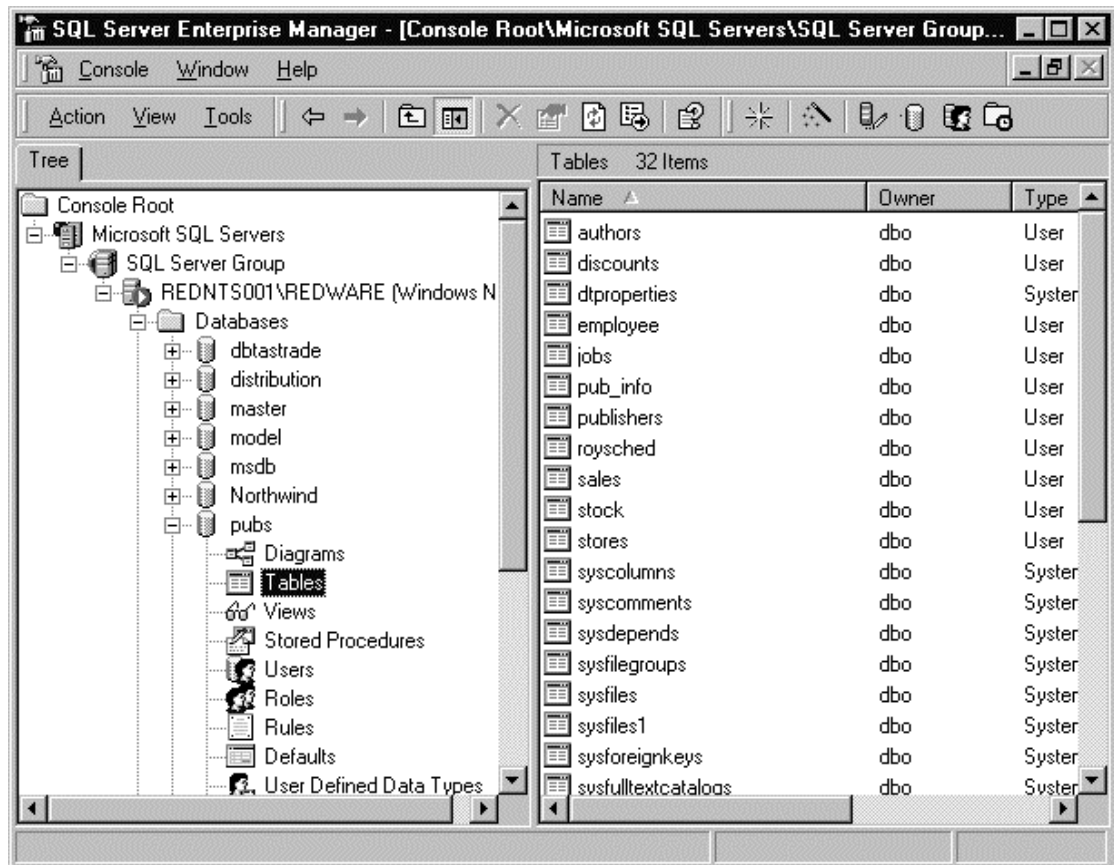


*Other services may also be controlled by the service manager by selecting from the Services combo.*

### Enterprise Manager

The SQL Enterprise Manager is the administrative interface for SQL Server allowing the access to all of the features of SQL Server. SQL Enterprise Manager utilises Microsoft's Distributed Management Framework (DMF) so that any SQL Server installation can be controlled from Windows 95 or Windows NT workstation.

After registering a SQL Server installation on the SQL Enterprise Manager, the Server can be controlled through use of the outline control. Clicking on the Server will bring up an outline which contains the objects controlled by the server.



### **SQL Server Enterprise Manager**

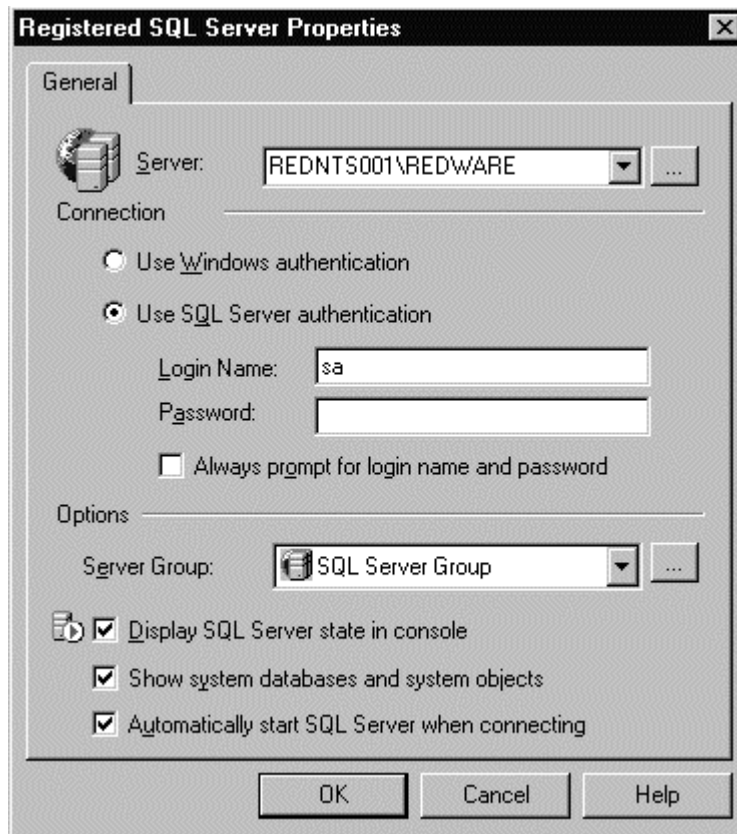
The Enterprise manager can be used to:

- Create a new database and alter database properties.
- Define new tables, views, stored procedures and user defined function within a database.
- Access tools such as the Query Analyser, SQL Profiler, and database diagrams.
- Define users and roles and set permissions on database objects.
- Schedule jobs with the SQL Agent.
- Set up and operate database replication.
- Import and Export data with the DTS.
- Perform database administration tasks such as backups.
- Monitor system activity.

### **Register the Server**

Before we can change the structure of a database we need to register the Server on the current workstation.

This happens automatically the first time the SQL Executive is run or an option can be selected from the Server-Register Server menu to add additional servers.



### Registering a Server

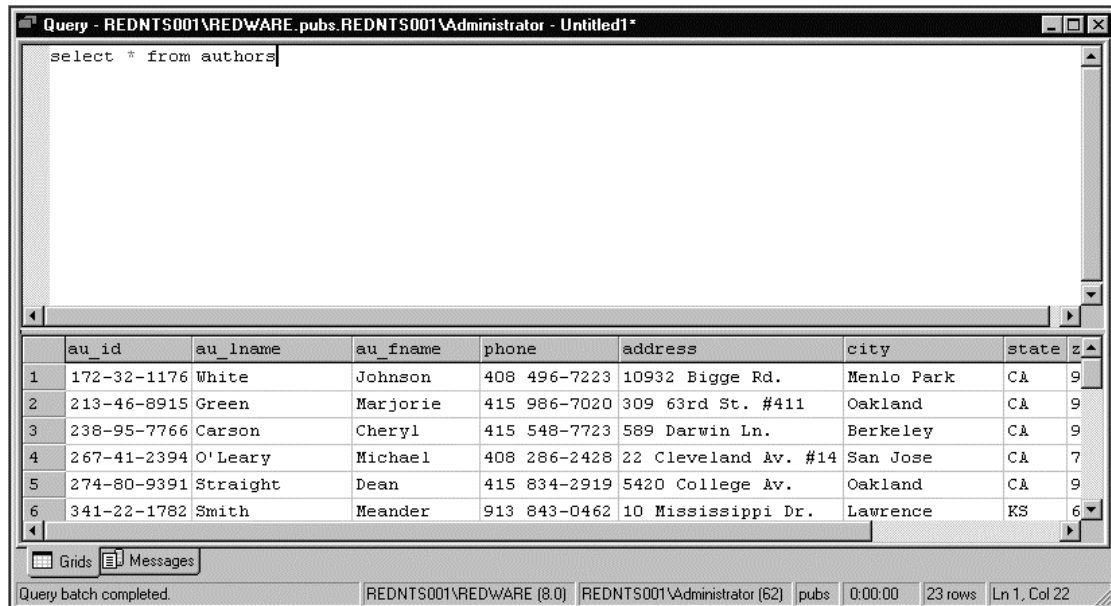
*There may be several separate SQL Server installations running on the same machine.*

## SQL Query Analyser

The SQL Query Analyser is available from the TOOLS.. menu of the Enterprise Manager. The user must log onto the required server and specify the required database using the Database combo at the top of the window. The user can then type in a SQL Server statement in the Query window and press the Execute button to see the results interactively.

The Query Analyser can be used to run any command that SQL Server understands as well as standard SQL queries. Simply and type in the required Transact-SQL commands and press the Green GO triangle icon to run the query and view the results. Note that there are two pages for results, one for results sets returned and the other for messages.

*Press SHIFT+F1 on any SQL keyword to launch SQL Books Online with the required information displayed.*

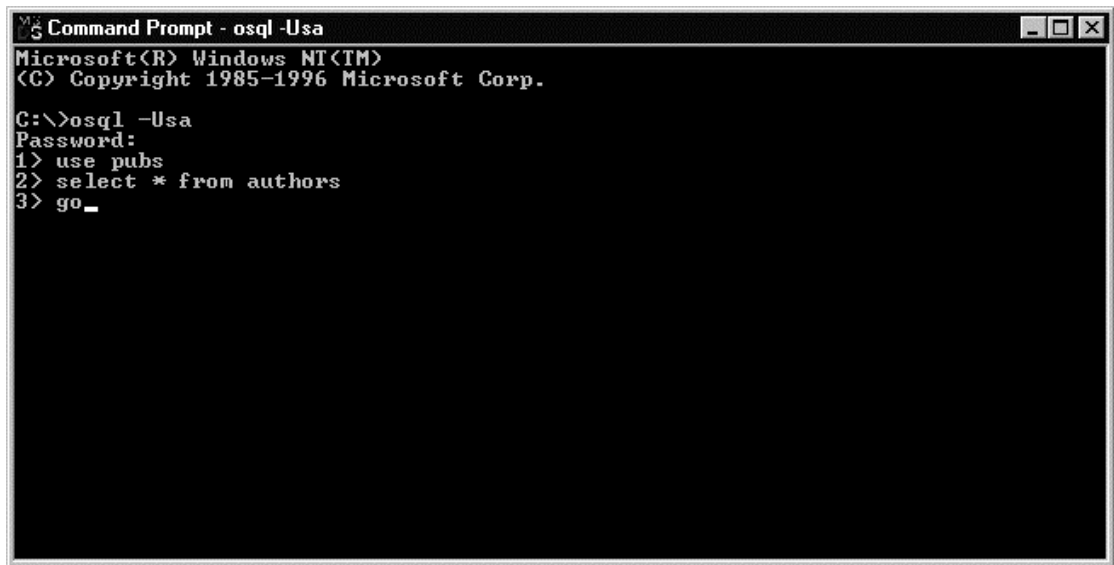
**SQL Query Analyser**

*Transact SQL commands can also be placed in a batch text file with a \*.SQL suffix and executed from the OSQL window by selecting the File-Open menu and then executing the query. A batch file may have several sets of SQL statements executed with a "GO" command.*

**OSQL**

OSQL is a command line utility which interacts with the database server from the DOS command prompt. It can be used to run jobs on the server from a windows standard batch file.

The following example logs onto the server as system administrator to query the authors table in the pubs database. The OSQL window is normally used to run a stored procedure.



```
Microsoft Windows [Version 5.00.2600]
Copyright (c) 2000 Microsoft Corporation

C:\>osql -Usa
Password:
1> use pubs
2> select * from authors
3> go_
```

## 4. SQL Syntax

SQL (Structured Query Language) has four main commands for manipulating data:

- SELECT existing data from one or more tables.
- INSERT a new record into a table.
- DELETE one or more records from a table.
- UPDATE existing records in a table.

This section explains these commands with particular reference to the flexibility of the SELECT command. The Query Analyser available from the TOOLS menu of the SQL Executive should be used to run the examples against the PUBS database.

### pubs

The pubs database is installed together with SQL Server and is used in most of the examples in this book. This small database contains data referring to book publishers and the titles they publish alongside details of the authors that write the titles and the stores that sell them.

TABLE	PRIMARY KEY	DESCRIPTION
AUTHORS	AU_ID	AUTHOR name and address details.
DISCOUNTS	STOR_ID	Discounts for each STORE.
EMPLOYEE	EMP_ID	EMPLOYEE detail with a link to the JOB description and the PUBLISHER employer.
JOBS	JOB_ID	JOB Descriptions
PUBLISHERS	PUB_ID	PUBLISHER name and address detail.
PUB_INFO	PUB_ID	Image and Text information on the PUBLISHER.
ROYSCHED	TITLE_ID	Royalty information for each TITLE.
SALES	STOR_ID TITLE_ID ORD_NUM	Detail record of the quantity ordered of each TITLE by a STORE.
STORE	STOR_ID	STORE name and address.
TITLE	TITLE_ID	Information on the Book TITLE.
TITLEAUTHOR	AU_ID TITLE_ID	Royalty percentage for each AUTHOR involved in a TITLE.

*A SQL batch program called INSTPUBS.SQL is installed with SQL Server to allow the reinstallation of a fresh PUBS database for training purposes. Please ask your system manager to install a new PUBS database if required.*

### SELECT Statement

The SQL SELECT statement is used to select a set of data from existing tables in the database. The syntax of the command is designed to define a set of data which includes the fields (columns) and the set of records (rows) which should be selected.

The structure of the command is as follows:

```
SELECT [ALL | DISTINCT]
      [TOP n [PERCENT] [WITH TIES]]
```

```
select_list
[INTO [new_table_name]]
[FROM {table_name | view_name}[(optimizer_hints)] [ ,...n ]]
[WHERE clause]
[GROUP BY [ALL] expression [ ,...n ] [WITH {CUBE | ROLLUP}]]
[HAVING clause]
[ORDER BY clause]
[COMPUTE clause]
[FOR XML [ AUTO | RAW | EXPLICIT ]
```

### Field List

The select list is the list of fields or expressions that are required in the selected table. These correspond to the fields in the result set:

```
SELECT au_fname, au_lname FROM authors
```

The asterisk can also be used to select all fields from a table.

```
SELECT * FROM authors
```

An alias can be given to the field name to rename the field in the result table:

```
SELECT au_lname AS surname, au_fname AS firstname FROM authors
```

Expressions can also be specified for a field expression:

```
SELECT au_lname + au_fname AS fullname FROM authors
```

Expressions can be used in the select list:

```
SELECT s.*, t.price, qty * price AS qtyprice
FROM sales s
INNER JOIN titles t ON s.title_id = t.title_id
```

Scalar functions can be applied to fields or expressions for more complex queries:

```
SELECT UPPER(au_lname), CAST(address + ',' + city + ',' +
state + space(1) + zip as varchar(45))
FROM authors
```

### WHERE Clause

The WHERE clause is used to narrow down the rows selected for the result table.

```
SELECT * FROM authors WHERE au_lname = 'White'
```

AND and OR and NOT can be used:

```
SELECT * FROM authors
WHERE (state = 'CA' or state = 'UT')
AND (NOT contract = 1)
```

The IN clause can be used instead of OR:

```
SELECT * FROM authors
WHERE state IN ('CA','UT')
```

The BETWEEN syntax can be used also to select between given values:

```
SELECT * FROM titles WHERE price BETWEEN 10.00 AND 29.00
```

NULL values can be identified in a WHERE clause:

```
SELECT * FROM stores WHERE zip IS NULL
```

*The TOP n clause can be used to limit the number of records returned from a SELECT command.*

### Wild Cards

Wild cards can be used in selection criteria, for example to select any AUTHORS containing the letter 'a' in the surname:

```
SELECT * FROM authors WHERE au_lname LIKE '%A%'
```

The LIKE syntax allows a wide variety of pattern matching templates. The above example utilises the % character as a wild card symbolising any sequence of characters. Angle brackets are used to define a range of characters. The following example to picks out book titles with an identifier beginning with a character in the range B to M:

```
SELECT * FROM titles WHERE title_id LIKE '[B-M]%'
```

The underscore character indicates any single character, # any single digit, angle brackets picks any single character within the brackets, and [^] will pick any character not within the angle brackets. This example selects Titles which do not have P or M as the first letter of the identifier and have 1 as the third character:

```
SELECT * FROM titles
WHERE title_id LIKE '[^PM]_1%'
```

An escape character can be defined to allow one of the wild cards to be used as a literal in the expression. This example finds any occurrence of the % character in the PAYTERMS field of the STORES table:

```
SELECT * from sales
WHERE payterms LIKE '%\%%' ESCAPE '\'
```

*SQL Server is often configured not to be case sensitive. You may need to check this option or use expressions of the form WHERE UPPER(au\_lname) = 'SMITH'.*

### FROM Clause

The FROM clause specifies which tables are involved in the SELECT statement. The clause is mandatory:

```
SELECT * FROM authors
```

If more than one table is required in the query then they may be separated by commas:

```
SELECT * FROM titleauthor, authors, titles
WHERE titleauthor.au_id = authors.au_id AND
titleauthor.title_id = titles.title_id
```

The tables may be assigned an alias if required to shorten or provide an alternative name in the statement:

```
SELECT * FROM titleauthor ta, authors a, titles t
WHERE ta.au_id = a.au_id AND
ta.title_id = t.title_id
```

The alias name also allows for a recursive query that uses the same table twice to show an employees manager for example:

```
SELECT employee.surname, manager.surname ;
      FROM employee, employee manager ;
      WHERE employee.manager = manager.id
```

SQL Server uses a fully qualified name to identify a table. There are four parts to the fully qualified name: **server.database.owner.table**. The FROM clause can specify a table that is located in another database or even another table:

```
USE northwind
SELECT * FROM pubs.dbo.authors
```

### ORDER BY

The ORDER BY clause allows for a result table to be ordered in any desired sequence:

```
SELECT * FROM authors ORDER BY au_lname, au_fname
```

The order sequence may also refer to the output fields in the result table using a number indicating the sequence of the field in the select list:

```
SELECT au_lname, au_fname FROM authors ORDER BY 2,1
```

The DESC keyword may be used to reverse the sort order of a column in the ORDER part of the SELECT statement:

```
SELECT title, ytd_sales, type as category
      FROM titles
      WHERE type = 'business'
      ORDER BY 2 DESC
```

### Natural Join

WHERE clauses are often used for joining tables together using the primary and foreign keys. Relational databases allow for any two tables to be joined together with an expression in the first table matching any expression in the second table as long as the width and data type of the expression is identical.

Joins allow considerable flexibility to the programmer in joining tables together although, nearly always, the programmer will want to join one table to another using the foreign and primary keys.

```
SELECT * FROM titleauthor ta, authors a, titles t
      WHERE ta.au_id = a.au_id AND
      ta.title_id = t.title_id
```

The more modern syntax for a natural join is to use the INNER JOIN syntax as follows:

```
SELECT * FROM titleauthor ta
      INNER JOIN authors a ON ta.au_id = a.au_id
      INNER JOIN titles t ON ta.title_id = t.title_id
```

*Specifying two tables in a SELECT statement without specifying a join condition will create a Cartesian product of both tables. This means that a 100 record table joined to a 200 record table with no WHERE clause will create a 20,000 record result table.*

### **GROUP BY Clause**

The GROUP BY command can be used with the aggregate functions to count up the number of occurrences of a value or to summate, average or perform statistical calculations on a table:

```
SELECT title_id, SUM(qty) AS totalsales  
  FROM sales GROUP BY title_id  
SELECT title_id, COUNT(*) FROM sales GROUP BY title_id
```

The following example shows the maximum, minimum and average order level for each title together with the total number of records for each title and a count of the number of different stores ordering the title:

```
SELECT title_id, COUNT(*) AS ordercount,  
  COUNT(stor_id) AS storecount,  
  MAX(qty) AS maxqty,  
  MIN(qty) AS minqty,  
  AVG(qty) AS avgqty,  
  SUM(qty) AS sumqty  
  FROM sales  
  GROUP BY title_id
```

The ALL keyword can be used to include all the groupings present in the table even if there are no occurrences selected in the query. The aggregated fields for the additional groups are set as NULL values:

```
SELECT title_id, SUM(qty) AS totalsales  
  FROM sales  
  WHERE YEAR(ord_date) = 1994  
  GROUP BY ALL title_id
```

The CUBE and ROLLUP options on the GROUP BY command are specific to SQL Server and add additional summary records into the selection. This can help in creating results sets for complex management report.

Another aggregation command is the COMPUTE BY clause and remember that the SQL OLAP manager provides full management reporting facilities.

### **HAVING Clause**

The WHERE clause filters the rows that are used in the query. The HAVING clause operates on a query that employs a GROUP BY clause but only after the grouping has been performed.

This allows the summary records to be selected on the basis of their aggregated values. The procedure is similar to performing a second WHERE selection on the final results table.

The following statement selects titles that have sold more than 50 copies:

```
SELECT title_id, COUNT(*) AS ordcount,  
  FROM sales  
  GROUP BY title_id  
  HAVING ordcount > 50
```

### **DISTINCT**

The DISTINCT clause is not often used but may be used to prevent duplicate rows from appearing in the results table.

The following command creates a results table with one record for each Title Type in the TITLES table:

```
SELECT DISTINCT titles.type FROM titles
```

*A similar result may be obtained with the GROUP BY clause.*

### Inner (Natural) Join

A natural join is the operation that joins tables together using a where clause or the more modern INNER JOIN syntax. The following statement will create a view that joins the TITLES and SALES tables:

```
SELECT t.title_id, t.title, SUM(s.qty) AS totalqty
FROM titles t
INNER JOIN sales s ON t.title_id = s.title_id
GROUP BY t.title_id, t.title
```

A natural, or inner, Join discussed above will only display records that qualify the join condition and that occur in both tables. A title that has no sales will not be included in the view.

### Outer Join

An outer join allows for records to be displayed from either table even if there is no corresponding record on one or other side of the join. The Outer Join may be a left or right outer join depending on whether all the records in the first or the second table are required. A full outer join will include all records from both tables.

The syntax for a left outer join between the TITLES and the SALES tables allows all TITLES to be displayed:

```
SELECT t.title_id, t.title, SUM(s.qty) AS totalqty
FROM titles t
LEFT OUTER JOIN sales s ON t.title_id = s.title_id
GROUP BY t.title_id, t.title
```

Missing values where there are no corresponding SALES records for a Title are represented as NULL values.

*A LEFT outer join includes all records from the table mentioned in the FROM clause, the RIGHT outer join includes all records from the joined table, and a FULL outer join includes all the records from both tables.*

### Sub Queries

Subqueries can be useful in creating views with complex selection criteria. The following example could be expressed as a normal JOIN and GROUP BY but is more clearly expressed as follows:

```
SELECT * FROM titles WHERE title_id IN
( SELECT title_id from sales
GROUP BY title_id
HAVING SUM(qty)> 25 )
```

Subselections are particularly useful when working with views or temporary sets of data and can be used to check if records are in or not in another table

```
SELECT * FROM titles t
WHERE title_id NOT IN
```

```
(SELECT title_id FROM sales s
WHERE t.title_id = s.title_id )
```

This same query could use the less specific NOT EXISTS clause:

```
SELECT * FROM titles t
WHERE NOT EXISTS
(SELECT title_id FROM sales s
WHERE t.title_id = s.title_id )
```

*A join may always be expressed as a subquery*

## UNION

The UNION command can be used to create a single view from two tables with a similar structure. The following example creates a single table from the authors and employee tables:

```
SELECT
  a.au_id AS cid, a.au_lname AS lastname, a.au_fname AS fname
FROM authors a
UNION
(SELECT e.emp_id AS cid, e.fname AS firstname,
e.lname AS lastname FROM employee e)
```

Duplicates are removed from the resulting query unless the UNION ALL keyword is specified.

*Care needs to be taken where the table structures are not identical. The CAST or CONVERT scalar functions can be used to change a data type in a SELECT statement.*

## FOR XML *mode* [, XMLDATA] [, ELEMENTS][, BINARY BASE64]

SQL Server 2000 allows for rapid creation of XML from standard select statements. This is useful in creating components that use XML to communicate information:

```
SELECT * FROM authors
FOR XML AUTO
```

The modes are as follows:

- AUTO defines an element with the same name as the table for each record and represents fields as attributes.
- RAW uses an element name <row> instead of the element named after the table.
- EXPLICIT allows precise definition of the XML tree.

XMLDATA specifies full data type information using an external schema.

The ELEMENTS clause is used together with AUTO to include the columns of the select statement as sub-elements instead of attributes in the XML.

```
SELECT * FROM authors FOR XML AUTO, XMLDATA, ELEMENTS
```

*SQL Server 2000 has additional commands that allow a stored procedure to read an XML file (OPENXML).*

## SELECT .. INTO

A new table can be created with a SELECT INTO command provided that the user has CREATE TABLE permissions on the database.

```
SELECT *
  INTO contractauthor
  FROM authors
 WHERE contract = 1
```

## INSERT Statement

The INSERT statement allows new records to be added into a table:

```
INSERT [INTO]
  {table_name | view_name} [(column_list)]
  {DEFAULT VALUES | values_list | select_statement}
```

The INSERT statement requires that the values satisfy any validation constraints specified on the table otherwise the transaction will fail.

```
INSERT INTO authors
  (au_id, au_lname, au_fname, contract )
  VALUES ( "999-99-9001", "Crook", "Stamati", 1 )
```

*There are constraints defined on the Authors table that will prevent a new record from being added if the identifier is not unique or if the firstname, lastname, or contract field values are not specified.*

The INSERT command may also be used in combination with a SELECT statement to add records into a table:

```
INSERT INTO bestseller
  ( title_id, qty )
  SELECT title_id, SUM(qty) FROM sales s
  GROUP BY title_id
  HAVING SUM(qty) > 25
```

## UPDATE Statement

The UPDATE statement allows values in existing records to be changed:

```
UPDATE {table_name | view_name}
SET [{table_name | view_name}]
  {column_list
  | variable_list
  | variable_and_column_list}
  [, {column_list2
  | variable_list2
  | variable_and_column_list2}
  ... [, {column_listN
  | variable_listN
  | variable_and_column_listN}]]
[WHERE clause]
```

The Update clause can be used to update any field and usually involves a WHERE clause. Take care to specify the WHERE clause carefully or all the records will be updated:

```
UPDATE authors
  SET au_lname = 'Crank',
      au_fname = 'Stanley'
  WHERE au_lname = 'Crook'
```

*The WHERE clause is often used in conjunction with the Primary Key expression to update a single record in the table.*

The UPDATE command can also set values into a table by making calculations using data from another table:

```
UPDATE sales
  SET qtyprice = qty * (SELECT price
                        FROM titles t WHERE sales.title_id = t.title_id )
```

## DELETE Statement

The DELETE statement allows for deletion of table records using a WHERE clause to specify the records for deletion:

```
DELETE [FROM] {table_name | view_name}
  [WHERE clause]
```

The DELETE statement must satisfy any referential integrity constraints set up in the database before records are deleted:

```
DELETE authors
  WHERE au_lname = 'Crook'
```

*The **TRUNCATE TABLE authors** command could be used to delete all the records in the table. Take care to backup after such a command because a Truncated Delete is not logged.*

## 5. Database Definition

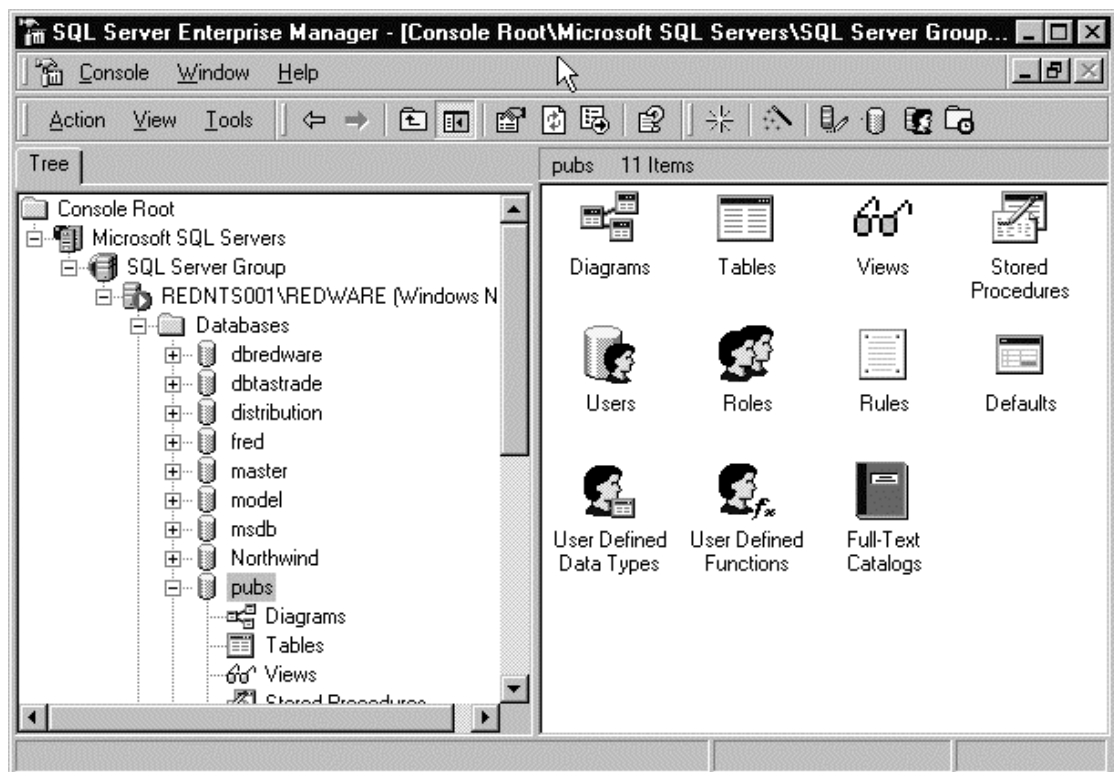
This section introduces the Enterprise Manager as a means of defining a database and its constituent tables. The following procedures are covered in detail:

- Create a new database and set database parameters.
- Create a Table and constituent fields.
- Define additional properties on a field.
- Add default and check constraints to a field.
- Define a primary key.
- Define a foreign key and set up referential integrity constraints between tables.
- Create a user defined data type.
- Generate a SQL Script for the database objects.

### Enterprise Manager

The Enterprise manager can be used to manage a number of SQL servers throughout the enterprise. Most options are made available by navigating the tree structure of the Enterprise Manager and right clicking on the desired option.

#### *Enterprise Manager*



Each installation of SQL Server has several system databases:

- MASTER contains many configuration details of the server including details of the schema for each database. It is important that this database is backed up whenever changes are made to the structure of any of the databases as it is very difficult to repair a system if the MASTER database is missing.