SQL-SERVER 7.0

KAPITEL1 – OVERVIEW

File-based Systems: Client Anwendung ist Client und Server (Interprozeßkommunikation), nur Datenkommunikation und Speicherung finden extern statt
Host-based Systems: Mainframes und Microcomputer, Zugriff über Terminal
Client-Server Systems: vom Client separierte Datenbankdienste, zentrale Verwaltung und Sicherheit, flexibles End-User Interface, Clients kommunizieren niemals direkt mit der Datenbank sondern immer mit dem Server welcher mit den physikalischen Daten umgeht
Heute: System-Mix
RDBMS: Relationales Datenbank-Management System, bestehend aus zweidimensionalen Tabellen (Spalten:Atrribute, Zeilen:Tupel), über Ansichten etc. werden Relationen manipuliert, um neue Beziehungen zu schaffen
Transact-SQL: spez. SQL-Version, Daten einsehen, suchen, verwalten, unterstützt **ANSI SQL-92** erweiterte Funktionalität
Unterstützte Server Plattformen: **9X** (Server als Applikation), **NT** (Server ist Dienst)
Unterstützte Client Plattformen: alle **WIN32bit API** OS, **Win3.1** und **DOS** nur via **SQL v.6.0/6.5**, **UNIX**, **MAC**
SQL unter NT: Sicherheit (eigene Sicherheit für nicht-MS-Clients), Multiprozessor (SMP), Interaktion mit Event Viewer (schreibt in alle **3** logs), Dienst (remote Start/Stop), Performance Monitor, Index-Server, Cluster-Server (**MSCS**, benötigt NT Enterprise Edition), integriert sich ebenfalls in die BackOffice Reihe (IIS, Exchange, SNA, SMS ...)

Clientkomponenten: Befehlszeilenoperierung, WIN32 GUI Utilities
2 Clientkommunikationskomponenten: DB Interface und Net-Library
4 Server-Dienste: **MSSQL-Server**, **SQL-Server Agent**, **MS Distributed Transaction Coordinator**, **Search Service** (Suchservice ist optional)

1. **MSSQL-Server Service**: RDBMS, File und Prozeßmanagement, Hauptkomponente, Koordination von Ressourcen und Usern, Konsistenz und Integritäts-verantwortlich, Sicherheitsüberwacher
2. **SQL Server Agent Service**: erstellen und verwalten von Jobs (lokal oder multiserver), Warnmeldungen, Operatoren
3. **MS DTC**: Transaktionsmanager, Integration von div. Datenlokationen in eine Transaktion, Two-phased-commit Replikation für permanente Updates, benutzt kompatiblen Standard **X/OPEN XA**
4. **Search Service**: ermöglicht Volltextsuche, optional installierbar

Clientsoftware: **Enterprise Manager**, **Query Analyzer**, **Admin Tools und Wizards**, **Command-Prompt Management Tools**

1. **SQL Server Enterprise Manager**: Administrations- und Datenbankverwaltungs-Client, benutzt **MMC** (MS Management Console, Schnittstelle für BackOffice Server Management)
2. **SQL Server Query Analyzer**: sendet individuelle oder Stapel-Befehle an Server, Abfrageanalyse und Statistiken, er ersetzt die frühere Version **ISQL /w**
3. **Grafiktools**: **Client Configuration**, **Performance Monitor** (settings file for preconfigured view), **Server Profiler** (capture and audit, früher: **SQL Trace**), **Service Manager** (Dienststart /Stop /Pause), **Server Setup** (Installation und Rekonfiguration), **Wizards**, **Data Transformation Services** (Import und Export von Daten zwischen heterogenen Quellen mittels **OLE**)
4. **Command-Prompt Management Tools**: führen SQL statements und script files aus, z.B. **osql** (um batch files auszuführen), **bcp** (export/import data zu/von SQL Server/nicht-SQL DBs, benutzt Text- oder Binärdateien)

Hilfesoftware: Application Help, Transact-SQL Help (über Highlight Funktion), SQL Server Books Online

Architektur:

Kommunikationsarchitektur: Schichtenarchitektur, so Applikations-Isolation von Netzwerk und Protokollen → Einsatz in unterschiedlichen Netzwerkumgebungen
Schichten vom Client: Application, Database Interface, Network Library

1. **Applications**: funktionieren über API, z.B. Enterprise Manager
2. **Database Interfaces**: Schnittstelle zu Applikation und Server → **OLE DB**, **ODBC**, **DB-Library**
3. **Network Libraries**: Kommunikationskomponente, gleiche Installation bei Client und Server erforderlich → **TCP/IP** (unterstützt nur 1 Protokoll), **Named Pipes**, **Novell IPX/SPX** (also auch Zugang für Novell Benutzer), **Banyan Vines/IP**, **Apple Talk ADSP**, **Multiprotokoll** (unterstützt mehrere Protokolle, Verschlüsselung, aber kein server name enumeration)
4. **Open Data Services (beim Server)**: Server Schnittstelle und **DLLs**

Datenzugriffsarchitektur: über **APIs** (Applikations-DB Schnittstelle, **2** Hauptklassen **OLE DB** (COM Schnittstelle, Component Object Model, benutzt sog. Provider wie SQL Server, Oracle, Jet, ODBC) und **ODBC** (TDS Protokoll arbeitet über Treiber)) und **Data Object Interfaces** (**2** Datenbankschnittstellen mit weniger Funktionalität als APIs: **ADO** (Active X Data Objects z.B. Visual Basic, Active Server Pages ASP und Scripting, einfacher zu implementieren als OLE DB!) und **RDO** (Remote Data Objects))
Administrative Architektur: **Clientseite** (Batch-Utilities, Enterprise Manager, COM Objects, SQL Distributed Management Objects (SQL-DMO, versteckt SQL Statements, zur Applikationsentwicklung)) und **Serverseite** (Server Agent (Warnmeldungen über Event Log, Benachrichtigung über Email, Paging und Anwendungsstart, Job Creation und Scheduling Engine, Replikation) und RDBMS) → Administration erfolgt grundsätzlich über SQL statements
Applikationsarchitektur: Database Design (model the business), **3** logische Schichten (Presentation Services, Business Services, Data Services)
4 Applikationsarchitektur-Schichtkombinationen: **Intelligent Server** (**Two-Tier** ~ server processing, Clients mit wenig Ressourcen, database-centric-point), **Intelligent Client** (**Two-Tier** ~ client processing, traditionelle Client/Server Umgebung, kleine Unternehmen mit Access z.B.), **N-Tier** (mehrere Server, logische und Datendienste separiert, multitiered enterprise applix), **Internet** (3 Schichten, keine Softwarewartung auf Client)

SQL Server Sicherheit:
2 Sicherheitsebenen: login authentication und permission validation
2 Login Authentication Mechanismen: SQL Server authentication und WINNT authentication, "**AUTHENTISIERUNGSPROZEß**"
SQL Server Authentication: kann eigene login accounts und pwds anlegen, wird bei SQL-Server Verbindung überprüft
NT Authentication: es kann über User- oder Gruppenkonto „grant SQL-Server access“ eingerichtet werden, dann keine erneute Überprüfung
2 Authentication Modi: **WINNT authentication Mode** (login nur über NT account, gibt es nicht bei Servern unter WIN9X) und **Mixed Mode** (login über NT oder SQL, Wahl liegt beim User), in früheren Versionen gab es auch noch **Standard Mode**, Modus muß spezifiziert werden

Berechtigungen: Aktionen werden vom SQL-Server überprüft, werden Userkonten und Roles eingeräumt
Datenbank User-Accounts: für eine spezifische DB bestimmt, kontrollieren Eigentum an Objekten, sind DB-spezifisch, "**MAPPING PROZEß**"
Database Roles: Roles==zusammengefaßte Usergruppen, es gibt server-level und DB-level roles, ebenso vordefinierte
Fixed Server Roles: sysadmin, DB creator, security admin ...
Fixed Database Roles: backup n restore, reading, modifying
Benutzerdefinierte R.: q.e.d.
MAPPING PROZEß: Nach Authentisierungsprozeß wird Login zu einem DB-Userkonto oder Role gemapped, ein Login kann nicht zu mehr als einem Userkonto gemapped werden, aber Mitglied verschiedener Roles sein!

→ Logins werden auf einen User oder mehrere Roles gemapped, sonst keinen DB Zugang !!!

KAPITEL2 – Installation

Hardware und Software Requirements: Intel oder DEC Alpha, min. **32MB** RAM, für NT Enterprise min. **64MB** RAM, HD Space **65/72MB** (Server only), **82/90MB** (Management Tools only), **170/175MB** (Typical), **180/183MB** (Full), NTFS oder FAT, NT Server od. Workstation 4 + **SP4** oder 9X, Internet Explorer **4.01** + **SP1**
3 Lizenzmodi: **PER SERVER** (simultane Verbindungen entsprechend vorhandenen Clientlizenzen werden am Server vergeben) oder **PER SEAT** (Vergabe an WKST und auch bei Benutzung von 3rd Party Software, gleichzeitige Verbindungen an Server unlimitiert), **INTERNET CONNECTOR LICENSING** (über IIS oder MS Transaction Server)
Wahl des Lizenzmodus: default wird PER SERVER gewählt, **1x** wechseln möglich, Replikation oder Data Transformation Services erfordert PER SEAT, Achtung WKST Verbindungen und USER Verbindungen sind unterschiedlich definiert !!!
Installationspfade: default **C:\Mssql7**, vorzugsweise **8.3** Ordnernamen wegen Befehlszeilenkommandos, **2** verschiedene Lokationen für Programm- (Root-Folder für Engines, Tools, vergrößern sich nicht) und Dateidaten (wachsen da Datenbanken, logs, backups, nach der Installation können weitere Pfade angegeben werden), ein Typical Install benötigt min. **72MB** auf dem Systemlaufwerk

Wahl des einzig-gültigen Character Set: Wahl der **CODE PAGE** der Sprachunterstützung (**256** Zeichen, Ziffern und Symbole, die ersten **128** bei allen gleich), gilt für alle Datenbanken auf dem Server, default **1252** (bei NT und UNIX/VMS Kompatibilität), weitere sind **850** multilingual (für internationale Unternehmen und DOS-Clients) sowie **437** (US spezifisch), Konfliktpotential wenn Client und Server unterschiedliche Character Sets benutzen (ebenfalls **REBUILD** von Datenbanken erforderlich wenn CS nach der Installation geändert wird), für Datenbankentwicklung in unterschiedlicher Sprache **Unicode** Datatypes benutzen (**65536** Zeichen-Standard); Character Set muß nicht gleiches wie NT benutzt sein
Übersetzungskonflikte: bei erweiterter Zeichenbenutzung wenn Client und Server unterscheidliche Code Pages benutzen
Code Page: speichert Codes, die Character Set mappen
Sortierreihenfolge: Vergleichsmechanismus bei Datensuche und Abfrageausgabe, nur für nicht-Unicode Daten, default **dictionary sort order case insensitive** (**AB = ab**)
Sortiereffekte: bessere Suchergebnisse und Performance, kein Referencing von Objekten
→ auf dem Server ist nur **1** Sort Order gültig für jede DB, Zusammenarbeit v. Servern benötigen für **BACKUP** einheitliche Sort Order, Datenaustausch mit verschiedenen SO unter Gefahren schon möglich, wird SO nach Install geändert, müssen ebenfalls DBs geändert werden
Unicode Collation: „Sort-Order" für Unicode, **1** lokaler (Priorität) und div. Vergleichsstyles, default **general case-insensitive width-insensitive Kana-insensitive**
Lokal: Regionen oder Länder werden z.B. vorrangig behandelt
Empfohlen: gleiche Sort-Order für Unicode und Nicht-Unicode
→ wird Unicode Collation nach Install geändert, ...**REBUILD**
Was tun wenn DB mit anderem CS als Server hinzugefügt werden soll: separaten Server installieren oder Unicode für neue DB benutzen

Default Network Libraries: **Named Pipes** (nicht WIN9X), **TCP/IP Sockets** (Port **1433**), **Multiprotocol** (Named Pipes, TCP/IP, NWLink IPX/SPX, andere **IPC** Mechanismen)
Beachte: Netzwerkprotokolle müssen von der Installation der Libraries vorhanden sein, NT Authentication braucht Named Pipes oder Multiprotocol, NT Verschlüsselung benötigt Multiprotocol und wird in der Registrierung von Server und Client eingestellt
Weitere Netzwerkbibliotheken: **NWLink IPX/SPX**, **Apple Talk ADSP**, **Banyan Vines** (unterstützt **SSP**)
→ Änderung nach Installation über SQL Server Network Utility

Server-Dienste-Konto: default user domain account oder system account als Sicherheitskontext

User Domain Account: Kommunikation mit **remote** Servern, benutzt **trustet connections** Sicherheitskontext, muß Mitglied der lokalen Administratoren sein („user must change pwd“ deaktivieren und „pwd never expires“ wählen), security, email notifications, Zusammenarbeit BackOffice Produkte, vertraute Domänen so wählen, daß alle User zugreifen können, PDC oder BDC install ungeeignet wg. Ressourcen
System Account: kann nur **lokale** Ressourcen ansteuern, z.B. NT PC nicht Domänenmitglied
Autostartservices: default Serverservice auto-startup (man muß sich nicht erst einloggen)

→ Server oder Komponenteninstall nur als lokaler Admin
3 **Installationstypen**: **TYPICAL** (Server, Mgm. Tools, Online Docu, no full-text search, no development tools, no code samples), **COMPACT** (s.o. aber keine Mgm. Tools), **CUSTOM** (z.B. nur Mgm. Tools → werden für Serverfernverwaltung gebraucht)
Was wird typischerweise installiert: **SQL Server Services** (Server, Agent, DTC, MS Search), **Mgm. Tools**, **DBs** (master, model, msdb, tempdb, pubs, northwind), **Folders and Files** (c:\mssql7\binn und ...\data), default **startup options** in Registry, default **security mode**: mixed mode, SQL Server **sa login** account ohne pwd, **SqlAgentCmdExec** Account (benutzt für jobs und services)
4 **Wege für Dienst-Handling**: SQL Server Service Manager, EM, Services in Systemsteuerung, Befehlszeile net command
Services unter NT: Dienst kann lokal oder remote gehandled werden
Dienstoptionen: **START** (Verbindungsaufbau, automatisierte Aktivitäten und Alerts nur im Startmodus des Agent), **PAUSE** (verhindert neue Verbindungen, Wartung, Nachricht kann an User verschickt werden), **STOP** (disconnect user)
Empfohlen: kein **STOP** über **Systemsteuerung** oder **net stop** Befehl, da so **checkpoints** vor dem OS shutdown nicht durchgeführt werden und die Recovery-Time beim Neustart ansteigt
Startoptionen/Parameter ändern über: ***Server-Properties-General Tab***
Serververbindung ohne Startupparameter über Befehlszeile herstellen: kaputte DB herstellen in **single user** mode (**sqlservr –m**) oder Konfigurationsprobleme beheben bei Startproblemen mit **Minimalkonfiguration** (**sqlservr –f**)
Einfache Serververbindung herstellen: (**osql –E –SSQLServer)**, dann SQL Befehle benutzen und zum Beenden **QUIT**

3 Unattended Installation Methoden:

1. **SAMPLE FILES auf CD**: via batch files (.bat) und setup initialisation scripts (.ins), Sql70**cli** (Mgm. Tools), Sql70**ins** (Typical), Sql70**cst** (Custom), Deske**ins** (Typical Desktop Edition), Deske**cst** (Custom Desktop Edition), Sql70**rem** (entfernt SQLServer)
2. **Eigene SCRIPT FILES**: 2 Möglichkeiten: (1) Server Setup interaktiv: Programm in **\x86\setup\setupsql.exe k=Rc** (generiert Skriptfile in \Winnt oder \Windows, cancel vor „Dateien kopieren“ im Installationsprozeß klicken, Sektionen [**SdStartCopy-0**] und [**SdFinish-0**] müssen noch hinzugefügt werden, wenn Server nicht tatsächlich installiert wurde beim Scriptfileanlegen, (2) mit Texteditor Datei mit Endung .iss anlegen, Setup starten über **start /wait setupsql.exe –f1 c:\ SQL7.iss –SMS –s**, **f1** weißt auf unattended install hin, **-s** steht für silent mode ohne User-Interaktion
3. **Microsoft Systems Management Server**: SMS 1.2 oder höher, automatisierte Installation auf vielen SMS-Computern, über Package Definition Format (**PDF**), Datei (**Smssql70.pdf**)

KAPITEL3 – Upgrading to SQL Server 7.0

Upgrade Process: **Server Upgrade Wizard** (**Data Transformation Services** oder **bcp** Befehl würden auch gehen um DBs zu verschieben, aber zu aufwendig)
Server Upgrade Wizard: entfernt nicht die 6.x Installation, nach dem Upgrade gibt es 2 unabhängige Installationen und 2 Daten-Sets, optional über „**TAPE UPGRADE OPTION**“ können 6.x devices aus Platzspargründen sofort entfernt werden
Zwischen Versionen switchen: über **Server-Switch application** über Startmenü oder **Vswitch.exe** in **C:\Mssql7\Binn** (nicht während Upgrade switchen!)

Upgrade Voraussetzungen: **NT4** mit **SP4** min. und SQL Server **6.X** mit **SP3** und SQL Server **7**, Named Pipes als Netzwerk-Bibliothek bei beiden installiert, auch bei tape upgrade (default pipe **\\.\pipe\sql\query**), SQL Server **4.2** muß zuerst in **6.5** und dann in **7.0** upgegraded werden, **1.5 x** soviel HD Platz wie **6.5** DBs, der Wizard schätzt aber auch automatisch (**DBs**, **LOGs**, **tempdb** Datenbank), bei Replikationsservern den **Distributor** zuerst upgraden
Weitere Upgrade-Tasks: (nach v7 Install und vor Wizard): **Database Consistency Checker DBCC** auf allen **6.x** Datenbanken und BACKUP, set **tempdb** DB mindestes zwischen **10MB** bis **25MB** (empfohlen), Logins in MASTER DB für DB user anlegen (jede upzugradende DB muß alle Objekteigentümer als Dbuser für Import enthalten), gespeicherte Startup Prozeduren deaktivieren, **2**-Computer Upgrade sollte über domain user account für Server Service ausgeführt werden, Replikationsstop, Replikations LOG muß leer sein und Serveranwendungen geschlossen
Upgrade der Master DB in 6.x: nur wenn Upgrade auf neuem Computer stattfindet: Servernamens-Referenzen in MASTER DB ändern, device file locations in MASTER DB updaten, alle Logins der zu transferierenden DBs in MASTER DB

Upgrade Wizard benutzen:
2 Upgrade Methoden: **One-Computer** (disk-to-disk named pipe oder tape upgrade) und **Two-Computer** (via named pipe connection, services brauchen domain user name und pwd auf beiden PCs in gleicher Domäne, der 6.x Server ist der **Export**-Server, 7.0 heißt **Import**)
2 Datentransfer Methoden: abhängig von Upgrade Methode und HD Space des Import-Server: **NAMED PIPES** (One-Computer, verläßlich, gute Performace da Upgrade im Memory erfolgt, freigegebener HD Space kann erst nach Beendigung des Upgrades benutzt werden), **TAPE BACKUP OPTION** (One-Computer mit limitierten HD-Platz, Bandgerät wird benutzt)
Weitere Option bei limitiertem Platz: alle **6.X** devices löschen (dann sollten aber alle DBs upgegraded werden, da dann alle **6.X** devices gelöscht werden)

Upgrade Steps and Options:
1. **DATA AND OBJECT TRANSFER**: **PERFORM EXHAUSTIVE DATA INTEGRITY VERFICATION** (Spalten **CRC**), siehe Log nach Fehlern
2. **CODE PAGE SELECTION**: Scripting Code Page muß spezifiziert werden (default MASTER DB entnommen), wenn default geändert wird dann besser kein Replikationsupdate!
3. **UPGRADE 6.X DATABASES TO SQL SERVER 7.0**: Datenbanken auswählen (MASTER, MSDB, DISTRIBUTION SYSTEM und SAMPLES nicht auswählbar, bereits upgegradete DBs sind nicht aufgeführt und in **EXCLUDED LIST**), DB Teile der nicht-aufgeführten DBs wie Logins, Konfigurationen, Tasks und Replikationseinstellungen können über gesonderte Optionen upgegraded werden, **verzweigt relationale DBs** gleichzeitig upgraden !!! Soll DB erneut geupgraded werden, dann erst wieder upgegradete aus **v7** löschen.
4. **SQL SERVER 7.0 DATABASE UND LOG UPDATE CREATION**: default automatically (Name, Pfad, Größe und Autogrow können angegeben werden) oder custom (eigene Konfiguration, über CREATE DATABASE neues v7 feature)
5. **SYSTEM CONFIGURATION**: **Systemobjekte** (upgrading von Server configuration, Replication settings, SQL executive settings: tasks), **ANSI Nulls** (columns default **NULL ON** (**NULL** oder **NOT NULL**) or **NULL OFF** (gibt **TRUE** oder **FALSE** zurück), important for comparisons) **QUOTED IDENTIFIERS** (Bedeutung der Anführungszeichen „“, **OFF** grenzt Zeichenstring ab → also **" ='**, **ON** grenzt einen Identifikator ab z.B. Spaltenname oder Namen mit ungültiger Schreibweise, Identifikator könnte aber auch optionslos mit eckigen Klammern abgegrenzt werden (wenn Settings später geändert werden ohne erneute Konvertierung gilt der ursprüngliche Zustand der Interpretation), **MIXED** bei Unsicherheit wählen oder bei unterschiedlicher Verwendungsweise → MIXED: Konvertierung aller Objekte zuerst zu ON, danach OFF Konvertierung der nicht funktionierenden Objekte)
6. **RUN THE UPGRADE**: Upgrade informiert während dem Prozeß über Upgradestadium

ANSI Anmerkung: in **6.X** werden ANSI NULLS bei Objekten bei der Ausführung aufgelöst, in **7.0** Auflösung bei Objekterstellung. ANSI NULL **OFF** bei alten Prozeduren die **SERVER NULLABILITY** benötigen, **ON** bei alten Prozeduren die **ANSI NULLABILITY** benutzen

SQL SERVER 6.X nach dem Upgrade entfernen: über Option im Startmenü, wird danach 6.X nochmal gebraucht, muß wiederum 7.0 ausnahmelos entfernt werden
Troubleshooting: Nicht alle Objekte können ohne vorherige Veränderung upgegraded werden
1. Textbezeichnungen in **syscomments** table müssen intakt sein: kein Upgrade bei Löschung, Umbenennung über **sp_rename**, Prozeduren mit Querverweis auf andere
2. Comutername muß mit Servername aus **@@SERVERNAME** übereinstimmen, korrigieren über **sp_dropserver** und **sp_addserver** Prozeduren
3. Gespeicherte Prozeduren, die die Systemtabelle verändern, oder auf nicht existente SQL 7 Systemtabellen oder Spalten verweisen, werden nicht upgegraded
4. Tabellen und Ansichten mit NULL-Name oder unzureichende Berechtigungen (Eigentümer-login ohne CREATE Berechtigung)

LOG DATEI: Upgrade Ordner in **C:\Sql7\Upgrade** mit Servername-Zeit-Datum wird jedesmal angelegt z.B. **SQLCONV1_092198_151900** (Bezeichnung_Datum_Uhrzeit), dort LOGs und Unterordner mit upgegradeten Datenbanken und **MASTER** DB, dort wieder LOGs mit Erfolgsendung **.OK** oder Fehlermeldung mit **.ERR**,
Kompatibilitäts-Level spezifizieren: wenn sich in der Version 7.0 features geändert haben um frühere Verhaltensweisen zu erhalten (werden für **USER ~** auto setzt Level auf Version des Export-Servers, **MODEL ~** default 7.0 und **MASTER** ~ default 7.0 DBs bestimmt), über Prozedur **sp_dbcmptlevel** mit Versionsangabe 6.0, 6.5 oder default 7.0
4-Level-Rückwärts-Kompatibilität (formelle Feature-Level-Gruppierung):
1. Entfernte Statements und Prozeduren in v7 (DISK REINIT, DISK REFIT)
2. Veränderungen mit Verhaltensmodulation (beim Updaten mehrerer Transaction Logs, letzter RESTORE mit WITH RECOVERY, davor mit WITH NORECOVERY)
3. Beibehaltene Items für Rückwärtskompatibilität aber nicht vorwärts (DBCC ROWLOCK, in v7 automatisch)
4. Geringfügige Verhaltensänderungen (JOIN Syntax)

(SQL-Server 6.5 kann auch mit **EM** registriert werden)

Zusammenfassung: nach Upgrade immer testen, manche Objekte müssen manuell verändert werden, nach dem Update alle Scripts in v7 updaten, auch wenn sie korrekt übertragen wurden
Beim 6.5 Upgrade beachten: erst SP3 installieren, TEMPDB min. 10MB, TAPE UPGRADE bei wenig Platz und devices löschen, davor 6.5 DBs backuppen (wegen device delete)
6.5 Prozedur kann nicht upgedated werden: modifiziert vielleicht Systemtabelle oder Querverweis auf nicht-existierende, Objekteigentümer nicht User der DB
Alte Abfrage benutzt cross als Spaltenalias, Kompatibilitätslevel v7: geht nicht, da **cross** in v7 reserviertes Wort ist, erst Umschreiben vor Komp.level v7!

KAPITEL4 – System Konfiguration und Architektur

Vorbereitungen um SQL Server zu benutzen
Enterprise Manager konfigurieren: Remote Server muß hier registriert werden, lokal wird auto registriert, die Registrierungsinfos sind entweder **privat** oder **shared** und befinden sich in der **Registry**
Registrierung: beim Client in der Registry nur, Information wird für Verbindung gebraucht (Registrierungs-Message poppt auto auf), benötigt [**Servername, Ntauth. od. SQLauth., Servergruppe**], bei Fehlermeldung bei Verbindung Registrierungsanfrage-Dialog
Register SQL-Server Wizard: mehrere Server mit **EM** registrieren, so Fernadministration möglich, sofern Mitglied der **sysadmin fixed server role** (lokale NT admins sind default Mitglied, sind globale dort eingetragen dann diese auch), Rechtsklick auf Gruppe um Wizard zu starten
Default Network Library EM: **Named Pipes**, über Client Network Utility kann diese geändert werden (da WIN9X können keine named pipes benutzen)
Server-Gruppen einrichten: Bei Registrierung wird Server entweder in die default **SQLServerGroup** plaziert oder es können neue Gruppen aus Organisationszwecken eingerichtet werden, diese sind allerdings unabhängig vom Server, sie können von Client zu Client neu bestimmt werden und haben keine Sicherheitsaspekte
Private Registrierungsinfo: default, andere können nicht darauf zugreifen

Shared Registrierungsinfo: **EM** Konfiguration gilt für viele User oder verschiedene Computer, **STORE USER INDEPENDANT** check box in ***EM-Tools-Options*** wegklicken (SQL Server Goup heißt jetzt Shared SQL Server Group)
Registrierung durchführen: ***Rechtsklick Servergroup-New Registration-Server*** oder ***[local]*** wählen, Authentizierung wählen und finish bestätigen
Client Installation: Management Tools können individuell ohne Server installiert werden, danach muß die Serverregistrierung erfolgen, gleiche Netzwerklibraries verwenden, bei 95/98 Client Network Utility benutzen um default named pipes zu ändern
SQL-Server Konfiguration:

1. **sa login account pwd** einrichten (über **EM** oder **sp_password** Prozedur)
2. **dynamic ressource managment**: auto ist effektiver als Administratorkonfiguration aber manuell kann man Verbindungen limitieren oder Speicher optimieren (über **EM** oder **sp_configure** Prozedur)
3. **default ANSI settings**: über **EM** oder **sp_dboption** kann das Verhalten einer Datenbank hinsichtlich ANSI Nulls, Quoted Identifier und ANSI Warnings eingerichtet werden (gilt für eine DB oder eine Verbindung). **SET** Befehl schaltet ANSI Verhalten für eine einzelne Verbindung aus. Optionen auf Verbindungsebene haben Priorität vor Datenbankoptionen !!!

zu (2) MEMORY SETTINGS: default **dynamic**, **fixed** (min/max server memory spezifizieren), **set working set size** (physikalischen Speicher reservieren, default **0** disabled, sonst **1**, i.V.m. max server memory sinnvoll bei anderen Ressourcen-saugenden Appliaktionen)
Troubleshooting:

1. **Cnfgsvr.out**: Setup Datei, speichert DBCC Database Consistency Checker Fehlermeldungen in **C:\Mssql7\Install**
2. **4 LOG Info**: bei jedem Start des Servers und Agentendienst neue LOGs, ebenso neue Event Viewer Einträge: **Sqlstp.log** (Info über Installationsprozeß in **C:\Winnt**), **NT Application Event Log**, **SQL Server Error Log** (Ereignisse in **C:\Mssql7\Log**, **EM** (***Server-Management-SqlServerLogs***) oder Notepad) und **Agent Error Log** (Warnungen und spezifische Fehler in **C:\Mssql7\Log**, **EM** oder Notepad)
3. **Netzwerkverbindungen testen**: wenn remote nicht zugreifen kann: **makepipe**, **readpipe**, **odbcping**, **ping** um Problemursprung zu identifizieren
4. **Gängige Probleme**: **Dienst kann nicht starten** (Dienste können nicht auf Domänencontroller zugreifen ~ Neukonfiguration oder System Konto benutzen /// falsche Registryeinträge ~ **regrebld** um Registry neuzubilden), **Error 1069 Logon Failure** (Änderung Pwd oder fehlende Berechtigungen ~ über ***Systemsteuerung-Dienste*** um neues Pwd zu spezifizieren und/oder Rechte überprüfen), **Management Tool cannot connect** (Service has to be started !!!), **Connection could not be established to XXX** (no matching Libs, no permissions ~ login with sa)

SQL-SERVER AKTIVITÄTEN

2 Kategorien: Datenbank implementieren oder Datenbank verwalten
Implementierung: **Design** (objects /apllication logic /relationships), **Creation** (Tabellen /Objekte /Integritätsmechanismen /Index /Sicherheit), **Performance Design** (Indexing, RAID, filegroups), **Planning Deployment** (Analyzing Workload), **Administrative Efforts** (Konfiguration, Monitoring, Managing Jobs and Alerts and Operators, Security, Backup)
Datenbankadministration: Install, Upgrade, Build DB, Datatransfer, Backup N Restore Strategy, User und Apllix Security, Automation of Tasks..., Monitoring and Tuning, Replication
Client-Server Tools: Administrationstools sind alles Clienttools, die mit MSSQL Server Service arbeiten, außer bei Startup Parameter interagiert man mit dem Server über TransactSQL (Server muß laufen), mache Operationen können nur durchgeführt werden wenn kein anderer User die Datenbank benutzt, Tools können sich auch gegenseitig behindern da dann ja **2** gleichzeitige Verbindungen laufen
SQL-Enterprise-Manager Komponenten: **MMC Toolbar** (shared BackOffice Komponenete, einheitliche Umgebung, hier können auch eigene Konsolen mit 3rd Party snap-ins entwickelt werden), **EM Toolbar** (Action Menu, View Menu, Tools Menu), **Console Tree** (in administrativen Serverkomponenten navigieren), **Details Pane** (wie Explorer, manchmal auch **Task-Pad-View**

== HTML PAGES) → **rote Zickzacklinie** neben dem Server heißt augenblickliche Verbindung besteht, **grüner Pfeil Server gestartet** !!! → **EM** hat in **v7** kein eigenes Query Fenster mehr!
SQL-Server Properties in EM:

1. **Connections Tab**: Maximum Concurrent User Connections: default **0** (means **32767** max.), Default Connection Options Checkboxes (configure ANSI defaults, Datenformatierungsoption)
2. **Memory Tab**: default dynamically control memory usage

SQL Server Query Analyzer: design and test Transact-SQL statements, batches, scripts, „text editor", color syntax, grid or free view, diagramming of logical steps (um Ressourcenverwendung zu optimieren), Index analysis
Query Analyzer Komponeneten: Title Bar (Server, DB, Login), Current DB, Query Pane (Editor), Results Pane (displays query results, div. Tabs (Register) **MESSAGE** (Fehleranzeige) **RESULT** (freier Text) **RESULT GRID** (nicht editierbar, bei mehreren result sets auch mehrere Register) **EXECUTION PLAN** (grafisches Diagramm), mehrere Fenster können im Query Analyzer geöffnet werden, jedes Fenster benötigt eine separate Verbindung mit unterschiedl. settings, exklusive Benutzung der DB während anderes Fenster DB benutzt ist daher unmöglich
Neue Query öffnen: (1) New Query vom Query Menü (2) Connect vom Dateimenü wenn unterschiedl. Login Zeugnissen und Voreinstellungen
Features: Querys als Datei speichern, Transact-SQL batch seperator (keyword **GO**, zur Benutzung unharmonischer Statements auf einmal), mehrzeilenfähig (statements müssen nicht in die gleiche Reihe), highlight code wird benutzt um statement als Teil des Gesamtcode zu testen, status bar publiziert Fortschreiten der Ausführung, **PARSE** (grammatische Zerlegung) vom Query Menü (keine Ausführung aber Meldung ob korrekte oder fehlerhafte statements)
Keyboard Shortcuts: Execute (**CTRL-E, F5**), Find (**CTRL-F**), uppercase (**CTRL-SHIFT-U**), lowercase (**...-L**), Textresults (**CTRL-T**), Gridresults (**CTRL-D**), HELP (**F1**), Statement Help (**SHIFT-F1**)

2 Datenbanktypen: **System-DBs** (interne Benutzung) und **User-DBs** (selber erstellt), beide sind strukturgleich
4 (5) System DBs: **MASTER** (login accounts, Prozeduren, Variablen, DB Lokationen über Pointer, Fehlermeldungen), **MODEL** (DB-Template, enthält Systemtabelle, ca. **1,5MB** nach Installation), **TEMPDB** (Temp. Storage Area, ca. **2,5MB** nach Installation), **MSDB** (Storage area for scheduling, job, alerts, events and backup n restore history, ca. **8,5MB** nach Installation), „**DISTRIBUTION**" (Replikationshistory und Transaktionen, existiert nur, wenn Server für Replikation konfiguriert ist)
→ Systemdatenbanken sollten maximal gelesen, nie aber empfohlenermaßen geändert oder gelöscht werden, nur neue Prozeduren der **MODEL** DB hinzufügen, wenn diese in allen neuen DBs gebraucht werden !!!
Benutzerdatenbanken: **2** Samples **PUBS** und **NORTHWIND**
9 Datenbankobjekte: **TABLE** (data storage), **DATA TYPE** (System- und benutzerdefinierte Datentypen), **CONSTRAINT** (Integritätsregeln, Standardmechanismus), **DEFAULT** („nichtdefinierter" Wert für Spalten, wenn kein anderer Wert da ist), **RULE** (Wert-Gültigkeits-Check), **INDEX** (schnelle Speicherstruktur), **VIEW** (Ansicht), **STORED PROCEDURE** (Befehle oder Batches werden nacheinander ausgeführt), **TRIGGER** (auto ausgeführte Prozedur bei Datenänderung durch Benutzer)
System DBs und Objekte: default versteckt, kann über Serverregistrierungsinfo "**SHOW SYSTEM...**" geändert werden
2 Referenzierungswege auf Objekte: **FULLY QUALIFIED NAMES** (**4** identifier, **server.database.owner.object**), **PARTIALLY SPECIFIED NAMES** (Führende Identifier können durch Arbeitskontext ersetzt werden, d.h. unwichtige Identifier durch Punkte (wenn zwischen Identifier) oder auf einer höheren Identifier-Stufe anfangen (z.B. **owner.object**))
Defaults bei Teilspezifizierung: Server ist default lokaler Server, Datenbank ist default aktuelle, owner ist default login ID der Verbindung (Roles können auch Objekteigentümer sein), ist man Mitglied der **db_owner** oder **db_ddladmin** role dann empfohlenermaßen **dbo** als owner benutzen

Systemtabellen: gespeicherte Information heißt **Metadata**, Sammlung von Systemtabellen in jeder DB heißt **Database Catalog**, **Systemcatalog** ist eine Sammlung von Systemtabellen in der **MASTER** DB
Häufige MASTER Systemtabellen: **sysxlogins** (login accounts), **sysmessages** (vorgefertigte system errors und warning Meldungen), **sysdatabases** (DBs auf SQL Server), **sysservers** (remote server), **sysdevices** (DB und Disk devices), **sysusers** (NTuser, Gruppen, SQLuser, SQLroles, auch in anderen DBs vorhanden), **sysobjects** (DB Objekte, auch in anderen DBs vorhanden)
Häufige MSDB Systemtabellen: **sysoperators** (persönliche administrative Infos, email, pager), **sysalerts** (q.e.d), **backupfiles**, **backupset**, **sysjobs**
Zugriff auf Metadata: empfohlen nur mit vorgespeicherte Prozeduren, Funktionen oder Schema Views wegen Datenverletzungsgefahr, also keine eigenen Scripts!
System-gespeicherte Prozeduren: beginnen mit **sp_prefix**
Prozeduren: **sp_help [object]** (Objektinfo), **sp_helpdb [DB]** (DB info), **sp_helpindex [Tabelle]** (Tabellenindexinfo)
Systemfunktionen: query Systemtabellen mit SQL, geben einzelne Werte zurück (z.B. DB_ID, USER_NAME, COL_LENGTH, STATS_DATE, DATALENGTH)
z.B. SELECT USER_NAME(10) → gibt Benutzernamen zurück
Information Schema Views: für Zukunftsversionen tauglich da ANSI Informationsschema, Eigentümer sind **information_schema** user, z.B. information_schema.table (Tabellenliste der DB), information_schema.columns (Spalteninfo), information_schema.tables_privileges (Tabellensicherheitsinfo), neu in **v7**, Syntax z.B. SELECT * FROM information_schema.tables
Wichtig: Metadata anschauen (über schema views, Prozeduren, Funktionen, selber besser nicht)

KAPITEL5 – Datenbankdateien

Datenbank erstellen: Datenspeicherstruktur (mindestens **1** Datendatei und **1** Transaction Log)
Datenbankdateien: es gibt keine DB devices zur Speicherung mehr, sie werden jetzt als Datei gespeichert
3 DB File Types: **PRIMARY DATA FILES** (Key-File, nur **1** pro DB, gewöhnlich mit **.MDF** Endung), **SECONDARY DATA FILES** (Daten und Objekte, die nicht in Primary sind, sind nicht unbedingt benötigte Dateien, es kann aber auch mehrere geben, gewöhnlich **.NDF**), **LOG FILES** (Transaktionsinfo zur Wiederherstellung, mindestens **1**, **.LDF**)
DB Datei-Considerations: Kopie der MODEL DB ist Minimalgröße der DB, **Pages** (aufeinanderfolgene **8KB** blocks, **128** pages gehen in **1MB**), 1-Reihenmaximum ist **8060bytes** (Reihe kann sich also nicht über Pages erstrecken), Reihenmaximum pro page ist **8094bytes**, Tabellen/Objekte/Indices werden als **Extent** gespeichert (**8** aufeinanderfolgene pages oder **64KB**), DB hat **16** extents pro **1MB**, Transaction Log ist default **25%** of data file size
Mixed Extent: **8** kleine Objekte teilen sich **1** Extent
Uniform Extent: Wenn Tabelle wächst bis **8** pages, benutzt sie uniform extent
Arbeitsweise der Transaction Log: (über **INSERT**, **UPDATE**, **DELETE** statements) Datenmodifikation durch Applix → pages in cache → jedes statement wird als write-ahead log erfasst vor tatsächlicher Änderung in DB → Ckeckpoint schreibt erfolgte Transaktionen auf Disk (start with **BEGIN TRANSACTION**, end with **COMMIT TRANSACTION**) *** bei Systemfail auto Wiederherstellungsprozess bei Neustart, Transaktionslogging kann nicht ausgestellt werden (nur bulk copy program **bcp** und **SELECT INTO** statement umgehen), TIP: disk-caching controller ausschalten zur Integritätssicherung !!!

3 Wege Datenbankerstellung: **DB Creation Wizard** (***EM-Servername-TOOLS-Wizards***), **EM** oder **CREATE DATABASE** statement
Achtung: MASTER DB immer backuppen vor create, modify, drop DB
Syntax und Optionen für CREATE DATABASE:

1. **PRIMARY Option**: Name der primary filegroup (enthält DB Systemtabellen), Startpunkt der DB der auf die anderen Dateien pointet, wird die Primary Option nicht verwendet, wird die erste erwähnte Datei die primary data file
2. **FILE NAME Option**: OS filename und path auf lokalem Server, SQL Installationsordner

3. **SIZE Option**: Initial Size der Dateien (MB oder KB suffix), min **512KB** (keine Angabe wird Größe der .MDF aus MODEL entnommen), bei nicht spezifiziertem Logfile default **25%** oder **512KB** (größere wird gewählt), secondary datafile und logs nicht spezifiziert dann default **1MB**, Minimumgröße über **DBCC SHRINKFILE** reduzieren
4. **MAXSIZE Option**: s.o., MB oder KB suffix, Maximalgröße der Dateien, wenn nicht spezifiziert dann wachsen diese solange bis HDD voll ist
5. **FILEGROWTH Option**: **0** = no growth, Wert kann in MB, KB oder % spezifiziert werden, default **10%**, minimum **64KB, „inkrementelle Vergrößerungsrate“**, nicht mehr genug Platz da dann + Vergrößerungsrate Platzbeschaffung

Zu (2) → Datenfiles und Transaktionsfile auf separate physische Disks wenn möglich legen (Performance) !!!

DROP DATABASE DB_NAME: endgültige permanente Löschung der Datenbank und disk files, durch Komma getrennt können mehrere DBs gleichzeitig gedropped werden
DROPPING-Anmerkungen: mit EM kann nur **1** DB auf einmal gedropped werden, mit SQL mehrere, wenn DB war default für login dann jetzt ohne, MASTER DB backup nach DROP (MASTER, MODEL, TEMPDB können nicht gedropped werden, MSDB schon)
DROPPING geht nicht wenn: während restore Vorgang, geöffnete DBs, in Replikation verwickelte Tabellen
MSDB nicht DROPPEN wenn: Benutzung von Server Agent, Replikation, Server Web Wizard oder DTS

Managing DBs
Manuelle Vergrößerung oder Verkleinerung von DBs: über **ALTER DATABASE**, **DBCC SHRINKDATABASE**, **DBCC SHRINKFILE**
DB Information sehen: über **EM** (Register: General,space Allocated, Tables and Indexes) oder **Prozeduren**
Prozeduren: **sp_dboption** (auflisten von Optionen), **sp_helpdb** (alle DBs auf Server, listet Name, Größe, Eigentümer, ID, Erstellungsdatum, Optionen auf), **sp_helpdb DB_NAME** (bestimmte DB, zusätzlich Einzelheiten über Data und Logfile), **sp_spaceused OBJEKT_NAME** (Speicherplatzangabe, inkl. Log Datei)
Datenbankoptionen festlegen: via **EM** (***DB-Properties-Options***) oder **sp_dboption**: **DBO USE ONLY** (nur Eigentümer darf benutzen), **READ ONLY**, **SELECT INTO/BULK COPY** (akzeptiert Operationen ohne Logging), **SINGLE USER** (nur **1** user gleichzeitig, z.B. Wartungsarbeiten), **TRUNC. LOG ON CHKPT.** (truncate, d.h. erfolgte Transaktionen werden aus Log entfernt, benötigt dann aber sicherheitshalber komplette Backups), **AUTOSHRINK**
Achtung: DB Optionen können auf einmal nur mit **1** DB festgelegt werden, sonst muß man die MODEL DB ändern!
3 Methoden, die Datenbankgröße zu erweitern: als grow automatically konfigurieren, manuell Maximum Größe verstellen, neue secondary data files hinzufügen
ALTER DATABASE: Statement, um DB auto zu vergrößern: Optionen: **MODIFY FILE** (Name der Datei und Option angeben, nur **1** Option kann auf einmal geändert werden, sonst mehrere ALTER DATABASE statements), beim Vergrößern wird bei Überschreitung **MAXSIZE** angepasst, **SIZE** ist Minimalgröße, man kann Datenbanken aber so nicht verkleinern (→ benötigt DBCC SHRINKFILE !!!)
Current Size vergrößern: Maximum wird ebenfalls heraufgesetzt
Zusätzliche Secondary Data Files: Data File Maximalgröße **32Terabytes**, Logfilemaximum **4TB**, einfach unter ***Database-Properties*** in **EM** hinzufügen
Transaktions Log erweitern: bei erhöhter Datenaktivität (durch lacking WHERE statements, Bilder, index changes), Notwendigkeit wird über „Monitoring“ festgestellt (**EM** oder Performance monitor), Erweiterung des Logfiles über EM oder ALTER DATABASE
Transaction Log Monitoring Objekte in Performance Monitor: **Percent Log Used Counter** (derzeit benutzter Platz), **Log Bytes Per Flush** (Pufferbytes bei Speicherlöschung), **Log Flushes** (Anzahl der Flushes), **Log Flush Wait Time** (in Millisekunden), **Log Flush Waits** (Anzahl der Transaction Commits, die geflushed werden sollen)

3 Wege DB shrinking oder auch nur einzelne Files:

1. **DBCC SHRINKDATABASE**, kann aber niemals unter Minimumsize shrinken, shrinkt alle Dateien in der DB, Logfiles haben verzögerte (deferred) Operation und können nicht gezwungen werden !!! (**target_percent Option**: freier Platz in den Dateien in Prozent nach dem SHRINK Prozeß, mit NOTRUNCATE Option werden pages an den Anfang der File defragmentiert → default geht free space an OS, mit NOTRUNCATE bleibt dieser in der Datei, TRUNCATEONLY Option bedeutet free space am Ende der Datei geht an OS (kein page-moving) und target_percent wird ignoriert), **Syntax**: **DBCC SHRINKDATABASE (database_name [, target_percent] [, {NOTRUNCATE | TRUNCATEONLY])**
2. **DBCC SHRINKFILE**, Minimumgröße der DB Datei reduzieren (**target_size** Größe in MB, nicht angegeben dann wird maximal geshrinkt, **EMPTYFILE** migriert Daten in andere Dateien der Gruppe, danach kann das FILE allerdings nicht mehr benutzt werden), **Syntax**: **DBCC SHRINKFILE ({file_name | file_id} [, target_size] [, {EMPTYFILE | NOTRUNCATE | TRUNCATEONLY}])**
3. **Option Shrink Automatically**

Achtung: DB darf nie unter MODEL DB Größe shrinken, DB und MASTER DB immer vorher backuppen, SHRINKDATABASE und SHRINKFILE sind verzögerte (deferred) Operationen

Datenbanken auf mehreren Festplatten: entweder **Filegroups** oder **RAID**
Filegroups: anlegen wegen Performanceverbesserung, FG sind named collections of DB files, jede DB hat eine default filegroup, man kann weitere hinzufügen; Tabellen, Indices, Text, Ntext und Image Data aus Tabellen können einzelnen FG zugeordnet werden, um Festplatten-beanspruchung bei heavy queries zu reduzieren, FG können auch separat gebackupped werden, Log Files gehören nicht zu FG !!!, eine Alternative ohne administrativen SQL Aufwand ist RAID (kontinuierliches Backup)
RAID 0: keine fault tolerance, Disk Striping ohne Parität, beste RAID-Performance
RAID 1: Disk Mirroring, minimiert downtime
RAID 5: Disk Striping mit Parität, exzellente Performance, eine Plattengröße Daten ist redundant, benötigt mehr RAM
3 Filegroup Typen: **PRIMARY FILEGROUP** (alle Systemtabellen, hält Primary Datafile und evtl. Secondary, muß richtig gesized werden, wenn automatic growth off oder disk space begrenzt), **BENUTZERDEFINIERTE** (FILEGROUP keyword in CREATE DATABASE oder ALTER DATABASE), **DEFAULT** (kann irgendeine sein, zuerst ist primary default, DBO kann ändern)
Filegroup Info Prozeduren: **sp_helpfile file_name** (physische Namen und Attributsanzeige der Files), **sp_helpfilegroup filegroup_name** (Namen und Attribute...)
Performanceverbesserung: bevorzugt RAID benutzen, Daten und Transaktionsdateien auf seperate Platten mit eigenen I/O Controllern, FG gut gegen disk drive contention, **SQL 7** bietet bei FG **proportional space allocation algorithm** (clevere Platzverwaltung), FG vereinfachen auch Backup

Kapazitätsplanung:
Größenschätzer für DBs: MODEL DB und System Tabellen, + Daten und Wachstum, Indices, Transaktionslog (**25%** der DB-Größe für OLTP online transaction processing Umgebungen)
Tabellendatenschätzer: errechnen über Reihenanzahl, Reihengröße, Anzahl der Reihen pro page und Gesamtzahl pages
Pageanzahl in Tabelle schätzen: Bytes pro Reihe über einzelne Spalten zählen (bei variablen Werten Durchschnitt) → Zusammenzählen und **9bytes** für overhead addieren → **8094** wird geteilt durch Reihengesamtbytes und ergibt Reihen pro page (abrunden) → Gesamtreihen durch Reihen pro Page ergibt pages (diese sind jede **8KB** groß, remember?)
Indices schätzen: schwierig wenn **2** page Typen vorhanden (leaf pages ~ Index und Key Values, 2. Typ binary search tree)
Clustured Index: indexed order of key values of index, nur **b-tree** index
Nonclusteres Index: separate Struktur speichert Kopien der **key values** mit **Pointern** auf die Tabellenlokation, **b-tree** und **leaf pages** (dort key values und Pointer)

Rechenoperationen S.133-135

KAPITEL6 – Transferring Data

Datenimport/-export: mit Tools, SQL statements oder mit APIs (z.B. Data Transformation Services Object Model) eigene Programme schreiben
Warum Datentransfer: Verlagerung auf andere Server, Kopie, Archivierung, Migration
Prozess: Import und Export zwischen Ursprung und Ziel, dazwischen optional Datenmanipulation
Warum Datentransformation: fehlende Werte ergänzen, konvertieren, übersetzen, addieren, Datenbewegung zwischen heterogenen Quellen etc.
Beispiele:
Datenformat ändern: **1** und **0** in **true** und **false** wandeln
Daten restrukturieren und mappen: Daten aus verschiedenen Quellen kombinieren
Daten konsistent machen: „**DATA SCRUBBING**“, z.B. 1 2 3 in Good Average Poor wandeln (Representation war vor der Wandlung nicht konsistent, vice versa heißt die ABC Firma oder a-b-c Firma ab jetzt nur noch ABC-Firma)
Daten validieren: auf ungültige Datenvorgänge checken
Tools für Datentransfer: **DTS Import/Export Wizard** (interaktiv DTS Pakete schaffen für homogene und heterogene Umgebungen), **DTS Designer** (Datentransfer und Workflowgenerierung, homogene und heterogene Umgebungen), **dtsrun utility** (DTS über command prompt, DTS Pakete ausführen als Teil von batch oder scheduled job), **bcp** (native SQL oder ASCII Dateien transportieren, von Tabelle in Datei oder umgekehrt, für Text-Files geeignet), **SQL-statements** (SELECT INTO; INSERT; BULK INSERT; BACKUP; RESTORE), **sp_attach_db** (Datenbank an anderen Server übertragen), **Replication** (Duplikate erzeugen, ein ongoing process), **Host Data Replicator** (Involviert Mainframes z.B. IBM DB2 i.V.m. SNA Server), es können auch **eigene Programme** entwickelt werden
→ DTS (erzeugt komplette Transformation für ein Ziel) ersetzt nicht Replikation (forward an viele Ziele, reagiert auf Änderungen)
Bulk Copy Benutzung ermöglichen: ***EM-Server-Databases-Properties-Select into/ Bulk copy Option*** checken
bcp Parameter: **/c** (nur chars) **/t“,“** (Komma Feldterminator) **/r\n** (Neue Reihe Reihenterminator) **/e c:\xxx.err** (Fehlerdatei) **/b250** (Batchgröße) **/m50** (Maximalfehlerzahl) **/Server** (Servername) **/Usa** (Username) → als **.cmd** batch speichern
→ bcp Befehle dürfen nur eine Line benutzten, **RETURN** verboten!

DTS
Überblick: Datenverkehr zwischen SQL Server und OLE DB, ODBC oder Textdatei, man kann Daten und Schemas (nur) zwischen heterogenen DBMS kopieren, Benutzerobjekte i.V.m. 3rd party Produkten schaffen, interaktive oder scheduled **data warehousing** und **data marts**, Benutzung von Fremdapplikationen möglich
DTS Besonderheit: Will man Trigger, Prozeduren, Rules, Defaults, Constraints oder benutzer-defineirte Datentypen transferieren, müssen beide Computer SQL Server **v7** haben
DTS Prozess: mit extensible Component Object Model können neue OLE DB Datenquellen und Bestimmungen, Aufgaben und Transformationen erzeugt werden, man schafft DTS Pakete und führt diese aus
DTS Pakete: beschreibt alle Transferarbeit und Transformationsprozeß, definiert 1 oder mehrere koordinierte Transformationsschritte (**STEPS**) mit unterschiedlichen Operationen, sie werden über **EM** oder **dtsrun** ausgeführt, DTS ist ein Mechanismus
Verwendung der Pakete: Speicherung in MSDB DB, zum Microsoft Repository gelinked, als COM-Datei abspeichern
OLE DB Datenzugriffsformate von DTS: ANY !!! z.B. DBs, spreadsheets, txt...
4 OLE DB Provider: Softwarekomponente, die OLE DB interface anbietet, SQL Server 7 bietet folgende an: **NATIVE OLE DB** (Excel, Access, Workgroup und enterprise DBs), **ODBC** (Oracle, Access, DB2), **ASCII TEXT FILES** (Benutzung des SQL Server DTS Flat File OLE DB Provider), **CUSTOMIZED** (3rd party)
DTS Data Pump: Infrastruktur, Hochgeschwindigkeits und In-Prozess COM Server, der Daten bewegt und transformiert, arbeitet mit **VBScript**, **Microsoft JScript**, **PerlScript** in DTS Paketen zusammen → erlaubt complexe **ActiveX Scripts**

DTS Tools: Wizards, Designer, dtswiz, dtsrun, DTS node in EM
DTS Import/Export Wizards: DTS Pakete werden definiert, kopieren, Scheduling, auch Querys oder distributed querys (über **Query Builder** im Wizard), Objekttransfer, Start über Programmgruppe oder Befehlszeile **dtswiz**
DTS Designer: grafischer DTS Paket Editor, Objekte packen und Workflow spezifizieren, wird ein neues Paket gemacht, so öffnet der DTS Designer über die MMC ein neues Fenster mit **2** Paletten für Transformations tasks und Datenverbindungen, über DTSD kann auch DataWarehousing stattfinden.

Datentransformation mit DTS:

Restrukturierung und Mapping von Daten: formatieren und modifizieren, einzelne Spalten in mehrere verwandeln
Mapping Datentypen: Attribute können für Ziel spezifiziert werden, Transformations-flags geben Kovertierungsmöglichkeit an, jede Datenbank definiert eigene Datentypen und Spalten und Objekt Namens Koventionen (DTS schlägt bestes Default vor)
Daten einfügen und separieren: **FILE-LEVEL** (Infokombination von div. Quellen in 1 Ziel oder 1 Quelle in div. Ziele),**COLUMN-LEVEL** (wie oben aber auf Spaltenebene)
Transformationsschritte definieren: z.B. SQL statement ausführen, über Data Pump Daten aus homogenen oder heterogenen Quellen transformieren, Scripts ausführen, externe Programme aufrufen, DTS Pakete empfangen und ausführen
DTS Pakete speichern: zum ändern, wiederbenutzen, schedulen, wird ein Paket nicht gespeichert wird es sofort ausgeführt
3 Speicherwege: **im Sql Server** (lokale Pakete werden in MSDB gespeichert, sind für andere Server zugänglich), **im Microsoft Repository** (Datenbank mit deskriptiver Info über Softwarekomponenten und ihren Beziehungen, div. Schnittstellen, macht Pakete zugänglich für andere Applikationen, Datenabstammung wird festgehalten, Pakete werden in MSDB gespeichert, Metadata kann ins Repository importiert werden, Ansicht nach Import über Metadata Browser), **als File** (für email oder File-Server, sie erscheinen nicht im EM (die anderen 2 schon), öffnen mit ***Rechtsklick DTS-All Tasks-Open Package***)
Paketsicherheit: Verschlüsselung → alles wird verschlüsselt bis auf Paketname, Beschreibung, ID, Version, Erstellungsname
2 Sicherheitslevel von DTS-Paketen: **OWNER PASSWORD** (Komplettzugriff, muß zur Verschlüsselung vorhandensein) und **OPERATOR PASSWORD** (Ausführen aber nicht editieren und ansehen, benötigt owner password)

Workflows definieren: über PRECEDENCE CONSTRAINTS, DTS Tasks können Prioritäten zugewiesen werden, Workflows bestehen aus Steps, Tasks und Precedence Constraints
Steps: kontrollieren Ausführung der Tasks, kann eine oder mehrere Precedence Constraints (vorrangige Ausführhemmschwelle mit **finish-start** Beziehung → Step B cannot start until step A finishes, es könnten ja mehrere Beziehungen involviert sein) haben, hat er keine wird der Task sofort ausgeführt. Steps haben Icons und Pfeile zwischen Verbindungen
Precedence Constraints Farbpfeile/Typen: **COMPLETION** (blau, warten auf success oder failure bevor destination step ausgeführt wird), **SUCCESS** (grün, source step muß erfüllt worden sein, bevor destination step ausgeführt wird), **FAILURE** (rot, step muß mit Fail-indikator abgeschlossen sein bevor destination step ausgeführt wird)
DESTINATION STEP: constraint Pfeil zeigt zum Step zu dem er gehört, bei Datentransformation zeigt er zur Ursprungsdatenverbindung des Steps
SOURCE STEP: vom Step zum constraint, bei Datentransformation zeigt Pfeil zum Ziel
→ die Begriffe Destination und Source gelten für Steps und Datentransformation
Step Execution: sequentiell, parallel oder kombiniert
Conditional Processing: **IF-THEN-ELSE** Beziehung
3 Task Priorities: kann spezifiziert werden, default priority DTS packet for steps, **IDLE**; **NORMAL**; **HIGH**
DTS Pakete ausführen: **dtsrun /SSQLServer /Usa /N"StudyNWind Product Totals"**, Name mit Anführungsstrichen da spaces, **/P** gibt pwd an wenn da, **/Usa** bedeutet login mit sa account

Scheduling DTS Pakete: wenn in msdb gespeichert dann mit Import/Export Wizard, oder über EM als Server Job i.V.m. **dtsrun** für files, sonst Rechtsklick auf Paket und **Schedule Package** wählen

Linked Servers: Referencing auf andere Datenquellen auf anderen Servern über nur eine einzige Verbindung oder sogar mehrere wiederum gelinkte Server referenzieren und zwischen ihnen Update oder Leseoperationen ausführen
Linked Server hinzufügen: benötigt OLE DB Provider (DLLs, diese müssen vorhanden sein) und OLE DB Datenquelle (gewöhnlich andere SQL Server), Verbindungsinformationen und Datenquellen müssen registriert werden, danach kann die Datenquelle mit einem einzigen logischen Namen angesprochen werden, linken und löschen über Prozeduren (**sp_addlinkedserver**) oder EM
Sicherheit bei Linked Server: **mapped login für user** hinzufügen (login name und pwd, nicht sinnvoll bei vielen usern), wenn kein mapped login vorhanden, dann entweder kein Zugriff oder der user wird auf ein angelegtes Generalkonto für viele gemapped oder Datenquellen sind evtl. nicht geschützt oder es handelt sich um **impersonation** (SQL Server hat gleiches login und pwd auf linked server, deshalb übernimmt der Home-SQL-Server den Prozeß)
Impersonation: kann gesondert angegeben werden
Linked Server Prozeduren: **sp_addlinkeserver** (hinzufügen), **sp_linkedservers** (Info über LS), **sp_dropserver** (Definition löschen), **sp_addlinkedsrvlogin** (login mapping hinzu), **sp_droplinkedsrvlogin** (login mapping löschen)
Fully-qualified Anwendung: bei Abfragen von linked Servern: **linked_server_name.catalog.schema.object_name**, catalog (DB name), schema (Eigentümer der Tabelle)

KAPITEL7- Web Publishing und Full-Text Indexing

Publikation: HTML Seiten von Tabellendaten aus SQL Server generieren über **Web Assistant Wizard** (**sp_makewebtask**), sogenannte „**Web Assistant Jobs**“
Web Assistant Jobs: für jeden wird Reihe in System Tabelle in MSDB angelegt (Jobname und HTML Lokationsdatei), für jeden job wird eine Prozedur mit gleichem Namen generiert (beim ausführen erzeugt diese ein HTML-file), soll der Job bloß einmalig stattfinden wird dieser im Anschluß an Ausführung gelöscht, manuell kann man jobs über EM oder **sp_runwebtask** ausführen, oder automatisch über **SQL Server Agent Job** oder **3 trigger** (handelt bei Datenänderung)
Diese HTML Seite enthält: Query Info (über Spalten aus einziger Tabelle, Textqueryeingabe, Prozedurangabe), Update Timing Info, Dateilokation, Formatierungsinfo
Automatische Web Updates: **SCHEDULING OPTIONS** (only one time when I complete this wizard, on demand, only one time at, at regularly scheduled intervals, when sql server data changes), SQL Server Agent muß laufen um automatische getimete jobs ausführen zu können, nicht der Web Assistant Job sondern der Server Agent Job wird scheduled, dieser wiederum aber benutzt den Web Assistant Job, Server Agent Job wird über **sp_makewebtask** angelegt, Web Jobs können nicht editiert werden, nur Server Jobs, wird Web Job gelöscht (z.B. **sp_dropwebtask**) , dann auch synonym der Server Job
Triggered Updates: 3 Triggers werden definiert in bestimmter Tabelle, Query Tabelle kann aber auch eine andere sein, Trigger reagieren bei UPDATE, INSERT und DELETE, Triggers werden entfernt wenn Web Task gedropped wird
HTML-Output-Lokation: lokal oder remote, Ordner muß schon existieren, default **C:\Mssql7\HTML**, anderer Ordner aber empfohlen
Formatierungsoptionen: **Template Datei** oder **Eigenspezifikation** (hier aber auch defaults für ausgelassene Angaben), nach der Generierung kann die erstellte Webseite mit ergänzenden HTML Tags erweitert werden (sollen Seiten benutzerdefiniertes Layout haben, muß man ein Template einrichten, da sie sonst beim ersten Update überschrieben werden)
HTML-Template Datei: normale HTML-Datei einrichten, Endung aber **.TPL**, **<%insert_data_here%>** zeigt Einfügeplatz der Tabelle in der Datei, **<%begindetail%>** und

<%enddetail%> geben Reihenlayout an mit <%insert_data_here%> tag für jede Spalte der Abfrage
Web Assistant Jobs: EM oder Prozedur um Jobs zu managen, in EM ansehen über ***Management-Web Publishing***, man kann aber nicht alle auflisten, Ausführung über ***Rechtsklick-Start Web Assistant Job***, oder **sp_runwebtask [[@procname =} `procname `][,[@outputfile =] `outputfile`]**, procname ist der Web Ass. Job, outputfile ist HTML Datei. Löschen über delete oder sp_dropwebtask ...(s.o.)

Volltextsuche: Character-basierte Abfrage ab SQL Version 7, funktioniert über **MS Search Service** (Indexing und Search Engine, gleiche wie IIS → dort heißt die engine **Index Server**), kann nur installiert werden wenn auch SQL Server auf NT Server installiert ist
Installation: über **Custom** bei Server-Neuinstallation oder **Server Setup** (in **select Components** alles unchecken bevor Full-Text gechecked wird)
Indexing: Search Service ist unabhängig von SQL-Server, sie kommunizieren nur zusammen !!! **Volltextindex** und **Server Index** sind unterschiedlich
Volltextindex: Index in Ordnern und Dateien im Dateisystem, **Catalog** genannt, default location **c:\Mssql7\Ftdata**, strukturiert keywords
Catalog: werden für ein oder mehrere Indices der DB benutzt, DBs können aber keine Indices teilen
MS Search Service: erweiterte Funktionen wie **proximity** oder **linguistic searches** in mehreren Fremdsprachen, Speicherung in Dateien verschiedener Typen und extern zu DBs, bei Textdaten **4KB** bis **2GB** groß, auch spreadsheets und Worddokumente etc. werden unterstützt
Abfragen: Grundanforderung ist das Bestehen mindestens eines 1-Spaltenindex für jede Tabelle, die für Volltextsuche registriert ist, Wörter werden mit Index Schlüsselwerten verbunden, erhält Server eine Anforderung für Suche, so sendet der SQL Server die Anfrage weiter an Search Service, der Ergebnisse zurückgibt.
Noise Words: „**a**“, „**is**“, „**the**“ werden als redunante Störwörter ignoriert, Listen für div. Fremdsprachen sind unter **c:\Mssql7\Ftdata\Sqlserver\Config** nach Installation des Search Service verfügbar, diese können mit Editoren ergänzt werden, treten aber erst in Effekt wenn die Indices durch nachfolgende Indizierung erweitert werden (**Population**).

Volltext-Dumps: Indices werden im Dateisystem gespeichert aber von der Datenbank verwaltet, nur **1** Index pro Tabelle, Population entsteht nur durch schedule (in regelmäßigen Intervallen) oder spezifischer Anforderung, eine oder mehrere Indices werden als **Catalog** zusammengefaßt, mehrere Catalogs für große Tabellen empfehlenswert
Wartung von Volltext-Indices: regelmäßige Population sicherstellen vor Implementierung, folgende Update Methoden sind vorhanden:

1. **Full Population**: alles neu, egal ob geändert oder nicht
2. **Incremental Population**: Beachtung nur von Änderungen (oder bei fehlender Zeitstempelspalte in Tabelle, bei neuen Spalten, bei neuer Struktur des Tabellenschemata)

Asynchrone Population: wegen Zeiteinsparung (Standard Index kürzer als Volltextindex) und weil Volltextindex weniger präzise als Standardindex ist
Volltextindices deaktivieren: (nur fulltext index metadata bleibt) für bestimmte Tabelle, Volltext Index bleibt ebenfalls bis nächste Vollpopulation bestehen, der Index kann allerdings nicht benutzt werden da bei deaktivierten Tabellen ist Mechanismus blockiert, bei Reaktivierung bleibt alter Index erhalten (Zustand: noch keine Neupopulation) und kann so für Abfragen benutzt werden
Volltextsuche einrichten: Search Service muß laufen, Volltextindices müssen für Tabellen eingerichtet werden
3 Search-Service Startarten: Volltextsuche Objekt in EM, über Server Service Manager-MS Search, **net start/stop mssearch**
Volltextindices einrichten: Administration über Full-Text-Indexing Wizard, Kontextmenüs in EM oder Prozeduren (wird Wiz mit markierter DB gestartet, beziehen sich die Einstellungen nur auf diese bestimmte DB)
Prozeduren: **sp_fulltext_database** (Initialisierung oder Entfernung Indexcatalog für DB), **sp_fulltext_catalog** (Catalog einrichten oder löschen und Indexing-Aktion starten/beenden),

sp_fulltext_table (für Indexing Tabellenmarkierung od. Revers), **sp_fulltext_column** (Spaltenpartizipierung an/aus), **sp_fulltext_service** (Eigenschaften des Dienstes und Catalog cleanen)
Info über Volltextsuche: über EM oder Prozeduren
Info über Prozeduren: **sp_help_fulltext_catalogs** (gibt ID, Name, Ursprungsverzeichnis, Status, Tabellenanzahl für best. Catalog an), **sp_help_fulltext_tables** (gibt Liste eingerichteter Tabellen für Indizierung an), **sp_help_fulltext_columns** (gibt Liste eingerichteter Spalten für Indizierung an)
Info über EM: ***Volltextcatalog Objekt in DB-Catalog***, Eigenschaftenfenster gibt an: Name, Lokation, physischer Name, Status, Item count, Catalog Size, Unique word count, last population date
Volltextabfragen: erweiterte Suchabfragen, mehr als LIKE Operator nämlich Kombination von Wörtern und Phrasen, gewichten query terms und geben außerdem Vergleichswerte gegenüber ursprünglicher Suche an
Transact-Prädikate und –funktionen: **S.195**

KAPITEL8 – Backup and Restore Overview and Strategy

Backupstrategie weil: versehensweise DELETE, UPDATE, Viren, Feuer, Erdbeben, Diebstahl
Backuphäufigkeit: hoch wenn OLTP (Online Transaction Processing System), niedrig bei geringer Aktivität oder Hauptaufgabe Entscheidungsunterstützung, Schedulen wenn bei nicht so starkem Updating, diverse DBs und Teile der DBs unabhägig konfigurieren
Warum noch Backup: um DB gleichzeitig zu kopieren
Synonymwörter in statements erhalten aus früheren Versionen: dump (backup), load (restore)

1. **Complete DB Backups**: Schema, Filestruktur, Daten (alle aktiven DatenPages → backup ist daher gewöhnlich kleiner da nicht-aktive ausgelassen), Teile der Transaktions Logfiles (enthalten Aktivitäten seit Backupstart)
2. **Transaction Log Backups**: nur diese Log, modifizierende Transaktionen seit dem letzten xxx-backup, danach wird inaktiver Teil gelöscht wegen Platz, können nicht bei ausgestelltem Log durchgeführt werden, Logs können nur mit gemeinsamen und dazugehörenden Backups wiederhergestellt werden (MASTER DB geht nicht)
3. **Differential Backup**: Datenpages, die seit dem letzten Komplettbackup geändert wurden, können nur wiederhergestellt werden gleichzeitig mit Complete Backup. Wird eine DB wiederhergestellt wird je das letzte komplette und differentielle Backup herangezogen (MASTER DB geht nicht)
4. **File oder Filegroup Backup**: wenn Zeit unzureichend für richtiges Backup, erfordert gleichzeitiges Transaction Log Backup

Wer darf Backup machen: sysadmin fixed server role (all DBs), db_owner fixed database role (this DB), db_backupoperator fixed database role (this DB), weitere roles können eingerichtet werden
Backups wo speichern: HD files (bei Netzwerk-File geht das Backup nicht über EM sondern BACKUP command), tape (Tape muß lokal sein, Backup kann aber woanders wieder eingespielt werden), named pipes devices (3rd party software)
Server Online Backup Process: DB checkpoint und Herausfinden der **log sequence number** der ältesten aktiven Transaktion Log → Pagewriting directly von HD (kein Cache) → schreibt Transaktions Log Records von der **LSN** ans Ende der Log
Nicht funktionierende Operationen während Backup: DB files einrichten/löschen, Index einrichten, nicht-geloggte Operationen durchführen, DB shrinken, automatic DB growth
→ läuft sowas kann Backup nicht gestartet werden und auch revers
Wenn Transaktions Log voll wird: SQL Server verbietet weitere Modifikationen, deshalb entweder (**1**) Transaction Log Backup oder (**2**) Clear mit BACKUP LOG statement mit WITH TRUNCATE_ONLY Option (dieses Backup läßt sich nicht wiederherstellen) oder (**3**) man setzt TRUNCATE LOG in CHKPT. DB Option auf true (Transaktionslogs enthalten aber nicht alle Transaktionen hier → beachte deine Strategie)
Wann soll Backup stattfinden: User und SystemDBs unterschiedlich je nach Aktion

Bei System DBs: regulär und vor Änderungsvorgängen, MASTER DB wichtig für Start von SQL Server, wurde das versäumt muß der **Rebuild Master** (**rebuildm**) ran
Rebuild Master: System DBs werden als Einheit rebuilded
MASTER DB backuppen nach: CREATE/ALTER/DROP DATABASE (automatic file growth egal, Dateien oder Filegroups hinzufügen/löschen nicht egal), Prozeduren mit login security (DB security wie permissions oder roles egal), add or drop server, wenn backupdevices hinzugefügt/gelöscht werden, DB Umbenennung, serverweite oder DB _options und _configure Prozeduren (Achtung: natürlich auch backuppen wenn eine EM GUI diese Prozeduren versteckt
MSDB DB backuppen nach: Änderungen Jobs, Alerts, Agentoperatoren, wenn versäumt dann System DBs rebuild und re-create oben genannte
MODEL DB: Backup bei Custom Model
TEMP DB: geht nicht
USER DBs backuppen nach: after creation, after creating indexes on large tables, after clearing Transaction Log, after performing non-logged operations (WRITTEXT/UPDATETEXT – es gibt aber auch WITH LOG Option – SELECT INTO/bcp wenn trunc.log chkpt. DB)

Backup Strategies:
DB Backup Strategy: kleine Dbs, wenig Änderungen oder schreib-geschützt, wenn Verluste hingenommen werden können
DB und Transaction Log Backup Strategy: sicherer
Differential Backup Strategy: Sicherheit weiter erhöhen: erst Komplettbackup, dann differentiell, dann Transaction Logs seit der differentiellen Sicherung → hier wird Recovery-Time gespart
File or Filegroup Backup Strategy: für große über mehrere files partitionierte DBs, kombiniert mit Transaction LOG, wenn Zeit ist rar
Performance Considerations: mehrere physical devices nutzen (parallel backuppen), tapes sind langsamer als disks, Aktivitäten am Server minimieren während Backup

KAPITEL9 – Backing up Databases

Backup Devices: haben **physischen** (via OS: heißen physical, temporary oder backup devices) und **logischen** (In Systemtabellen gespeichert, heißen logical, permanent oder named backup devices) Namen, beim Sicherungs/Wiederherstellungsvorgang kann man logischen oder physischen Namen benutzen
Named Backup Devices: haben kürzere Namen, Einrichtung über EM oder **sp_addumpdevice**
sp_addumpdevice: disk, tape oder direct data to named pipe, beachte physisches Gerät ist erst erstellt nach erster Benutzung (dann wird **C:\Mssql7\Backup\Mydev.bak** erstellt), logische und physische Namen werden in **sysdevices** eingetragen, Backupdevices im Netz über **mapped drives** oder **UNC**
Syntax: **sp_addumpdevice [@devtype =] 'device_type', [@logicalname =] 'logical_name', [@physicalname =] 'physical_name'**, device_type ist **DISK/TAPE/PIPE**
Named Backup Devices löschen: über EM oder **sp_dropdevice**, über EM kann disk-device file nicht entfernt werden, nur manuell oder i.V.m. **DELFILE** Option
Temporary Backup Devices: bei one-time-only backup, wenn bestehende Loaktion gewählt wird so wird aus dem Temporary BD ein Named BD erzeugt, Temp. BD wird mit phys. Namen in **BACKUP** statement initialisiert
Device initialisieren: bei Erstbenutzung, wird mit physischem Namen initialisiert
BACKUP Syntax: **BACKUP DATABASE database_name TO { backup_device_name| {DISK|TAPE|PIPE} ='temp_backup_file' [, ...n]**
Mehrere Backup Devices benutzen: „**striped backup set**", Zeiteinsparung, alle involvierten Geräte müssen vom selben Media-Typ sein (z.B. Disk), müssen aber nicht die gleiche Größe oder Geschwindigkeit haben, Kombis aus phys. Und log. Geräten erlaubt, Restore braucht nicht die gleiche Anzahl Geräte wie Backup
Media Sets: Sammlung von BDs um 1 oder mehrere Backup sets zu enthalten, kann auch einzelnes BD sein, BD ist einzelne Datei wenn BDs in multi-device media set sind disk drives, bei tapes: ein oder mehrere tapes sind ein BD → zusammen heißen diese **media family**, das erste tape heißt **initial media**, alle weiteren **continuation media**

Media set considerations: Media Set Tapes können nur vom SQL Server benutzt werden, werden BDs als Mitglieder eines Media Sets definiert so können diese auch nur gemeinsam genutzt werden, will man ein einzelnes dennoch benutzen so muß das BD reformatiert werden, dabei werden allerdings die Daten auf den anderen Media Sets unbrauchbar

Backups durchführen: mit EM (Backup Wizard) oder BACKUP statement
Syntax: **BACKUP DATABASE {database_name | @database_name_var}**
TO <backup_device> [, ...n]
[WITH
[[,] FORMAT]
[[,] {INIT|NOINIT}]
[[,] RESTART]
[[,] MEDIANAME = {media_name | @media_name_variable}]]
Medianame Option: Media Set angeben, ist es ein neues BD oder neu formatiert, so wird dieser Name in den **Media Header** geschrieben, bei subsequenten Backups wird nur der Name überprüft, bis zu **128** Chars lang bei BACKUP statement, in EM bis **64** !!!
INIT|NOINIT Option: default NOINIT: backup an bestehendes BD anfügen, INIT schreibt backup set an Anfang des device und überschreibt alles außer dem Media Header
Backup Overwrite fails wenn: wenn **EXPIREDATE** für backup sets noch nicht abgelaufen sind, wenn Parameter mit Media Name falsch zusammenpassen, oder wenn man versucht ein Mitglied eines anderen Media Sets zu überschreiben
Format Option: overwrite BD und Media Set splitten → neuer Media Header → BD contents werden auch überschrieben, Achtung: FORMAT bietet kein Namenscheck !!!
Restart Option: start interrupted backup from point of interruption

Backup auf Tape Device:
Gespeicherte Backup Info auf Tape Label: DB Name, Time, Date, Backuptype
Standard Backup Format: **Microsoft Tape Format (MTF)**, SQL-Server und nicht-SQL-Server Backups können sich auf dem gleichen Tape befinden, man kann jedoch nicht auf von anderen Applikationen erzeugten Backups zugreifen
TAPE-spezifische Optionen: **UNLOAD** (default, zurückspulen und Kassette ausfahren), **NOUNLOAD** (Option solange bis was anderes angegeben wird), **BLOCKSIZE** (Größe in bytes ändern, wenn mit FORMAT oder SKIP und INIT benutzt), **FORMAT** (auf alle Volumes header schreiben → INIT und SKIP Optionen sind bereits in FORMAT enthalten, checkt nicht header oder pwd der Media), **SKIP** (ignoriert label, kann so nicht expiration date feststellen oder Schreibberechtigungen erzwingen), **NOSKIP** (ANSI tape labels lesen, default)

Backup LOG Syntax: **BACKUP LOG {database_name | @database_name_var}**
TO <backup_device> [, ...n]
[WITH
[[,] {INIT|NOINIT}]
[[,] [NAME = {backup_set_name | @backup_set_name_var}]]
→ ersetzt **DUMP TRAN** aus vorigen Versionen !!!
weitere Backup Log Optionen: …**[WITH... TRUNCATE_ONLY** oder **NO_LOG** (Transaction Log löschen)
Wann sollte Transaction Log gelöscht werden: vor Komplett DB Backup (danach sollte allerdings direkt BACKUP DATABASE ausgeführt werden), Anwendung der Optionen zerstört aber Transaction Log Backup Sequenz, daher muß vor fortführenden normalen Log backups die DB gebackupped werden, beide Löschoptionen können nicht auf einmal angewandt werden (sie sind synonym), Log file wird aber so nicht kleiner sondern nur Platz in der Datei wird freigemacht
NO_TRUNCATE Benutzung: wenn Daten, auf keinen Fall aber Teil der Primary Filegroup, kaputt sind, und Transaction Log erhalten blieb

KAPITEL11 – LOGINS, USER ACCOUNTS, USER ROLES

3 Datenzugriffsverifizierungen von SQL Server:

1. Authentication mit Login Account (NT oder SQL Server)
2. DB Zugriff für bestimmtes User Konto oder Role Membership
3. Zugriff auf Objekte oder Aufgaben (Tasks) über Berechtigungen an User Account oder Role

Login Accounts können sein: WINNT User Account, NT Gruppenaccount (User ist Mitglied), benutzerdefiniertes SQL Login Account, default SQL Login Account
→ werden in der MASTER DB, **sysxlogins** System-Tabelle gespeichert, bzw. Syslogins View
Default DB: wenn login SQL Server hinzugefügt, kann default DB gewählt werden, diese stellt den default Kontext für Aktionen des Users dar (Art „Homedirectory"), bedeutet aber noch keinen Zugriff (benötigt berechtigten User, Ntgroup, Role), gibt es den **default guest user**, kann dieser für Zugriff auf default DB benutzt werden, wird kein default angegeben, so ist es die **MASTER DB**
2 Default Login Accounts: **sa** (System Administrator, hat alle Rechte im Server und allen DBs) und **BUILTIN\Administrators** (default WINNT login für NT Administratoren, ebenfalls alle Rechte im Server und allen DBs)
NT Account SQL Zugriff ermöglichen: über **EM** oder **sp_grantlogin** (nur **system** oder **security Administratoren** können Zugriff herstellen)
Syntax: **sp_grantlogin 'login'**, login ist z.B. 'MOONSHEA\Paul', login für User oder Gruppen, kombiniertes Domänen/Usernamen Character Limit: **128** Zeichen
Guidelines um NT logins zu SQL hinzuzufügen: einziges Gruppenlogin unter NT ermöglicht, daß keine Änderungen in SQL notwendig sind bei Gruppenmodifikationen und verhindert so **orphane Objekte**, wird NT Gruppe gelöscht bleibt sie im SQL Server erhalten, NT User Accounts in SQL nur anlegen wenn dieser keiner Gruppe zugeordnet ist (besser Kollektivlösung), SQL Server kann einzelne NT-Gruppenmitglieder über **SUSER_SNAME** Funktion identifizieren
Löschvorgang von Accounts: erst unter NT, dann **sp_changeobjectowner**, dann SQL login entfernen
Orphane Objekte: deren Eigentümer nicht mehr in SQL Server existieren
Login-Management-Prozeduren: **sp_revokelogin** (NT login aus SQL entfernen), **sp_denylogin** (Ntlogin hindern sich mit SQL zu verbinden)
Passwortänderung: User kann jederzeit ändern über **sp_password**, Administratoren jederzeit für jeden über **EM** oder **sp_password** mit **NULL** als altes Pwd
SQL Login hinzufügen: über **EM** oder **sp_addlogin** (SQL login einrichten, nur **System** oder **Security Adminstrators**)
Syntax: **sp_addlogin 'login' [, 'password' [, 'database']]**, logins und pwds bis **128** Zeichen inkl. Buchstaben, Symbole, Ziffern, jedoch kein backslash, reservierte login Namen (sa, public) oder leerer String **NULL** oder **''**

---

2 Authentication Modi: NT Authentication Mode (in SQL 6.5: Integrated Security) oder Mixed Mode, (in SQL 6.5: Standard Security gibt es in 7.0 nicht mehr)
NT Authentication: Eingabe Loginname und PWD → Validierung und Herstellung der Sicherheitsattribute → bei Verbindung öffnet Client eine **Trusted Connection** und Sicherheitsattribute werden ohne weiteres an SQL Server gegeben → SQL Server checked NT login gegen SQL account in **sysxlogins** Tabelle auf Übereinstimmung → Verbindung wird akzeptiert bei Übereinstimmung (SQL Accounts speichern die **NT-SIDs Security Identifiers** und vergleichen diese über die Sicherheitsattribute)
→ befindet sich SQL Server in **Trusted Domains**, reicht das Login in einer Domäne für Zugriff aus, die meisten grafischen Tools benötigen keine Login- und Pwd-Eingabe, Befehlszeilen Utilities erlauben erweiterte Optionen für **Trusted Connections**, **WIN9X** erlaubt keine NT Authentizierung
SQL Authentication: Client stellt bei Verbindung eine **Nontrusted Connection** her und übermittelt SQL-Loginname und Pwd → SQL Server verifiziert über **sysxlogins** Tabelle → SQL erlaubt bei Übereinstimmung Verbindung
Authentication Modus auswählen: es gibt kein Modus, der nur SQL logins erlaubt, abhängig von Sicherheitsanforderungen und Netzwerkumgebungen

Vorteile NT Authentication: wenn alle Clients Trusted Connections unterstützen, ermöglicht mehr Features wie sichere Validation, Passwortverschlüsselung, auditing, Passwortablaufdauer, minimale Passwortlänge, Account Lockout, Gruppen können einfach hinzugefügt werden, schnelle Verbindung
Vorteile Mixed Mode: für Clients mit Non-Trusted Connections oder Internet Clients oder Mixed Client Groups, zusätzliche Sicherheitsebene über NT
Implementierung der Sicherheitsmodi: SQL Server Network Utility überprüfen ob Protokoll für Trusted Connections für NT Clients da ist, Rechtsklick Server in EM – Properties – Security Tab (SQL Server und NT Authentication wählen) – SQLServer Service Stop und Restart – NT Gruppen und User hinzufügen – **EM** oder **sp_grantlogin** um SQL Zugriff zu gestatten – **EM** oder **sp_addlogin** für SQL Server Login Erstellung

---

Datenbankzugriff: nach erfolgreicher Verbindung mit Server, Zugriff erhält man über User Accounts oder Roles die für jede DB separat verwaltet werden
Logins DB-Zugriff erlauben: über **EM** oder **sp_grantdbaccess**, nur gestattet von **DB owner** und **database access administrators**, so Eintrag in **sysusers** Tabelle in der betreffenden DB
Syntax: **sp_grantdbaccess 'login' [,'name_in_db']**, name_in_db ist optionaler Name für Account in der DB
Prozeduren für DB Zugriffsverwaltung: **sp_revokedbaccess** (Zugriff für DB entfernen), **sp_change_users_login** (Beziehung zwischen SQL login und SQL user in dieser DB ändern)
2 Default User Accounts von DBs: **dbo** (sa und sysadmin role eingemapped, alle Objekte gehören dbo, wenn von ihm geschaffen, kann nicht gelöscht werden), **guest** (wird benutzt wenn user keinen gesonderten Zugriff auf DB hat und DB guest user account besitzt, guest kann beliebig gelöscht/hinzugefügt werden außer MASTER und TEMPDB, default keine Berechtigungen aber Mitglied der public role)
Logins zu Roles hinzufügen: (ersetzen **Aliases** und **Gruppen** aus 6.5) es gibt fixed server roles und selbsterstellte roles (z.B. für Business Zwecke)

---

7 Fixed Server Roles: **Sysadmin** (kann alles, Äquivalent zu **sa**), **Serveradmin** (Serverweite Konfiguration), **Setupadmin** (Replikationsinstall und erweitertes Prozedurmanagement), **Securityadmin** (Loginsverwaltung), **Processadmin** (Prozeßverwaltung auf Server), **Dbcreator** (DBs einrichten und verändern), **Diskadmin** (Disk Files verwalten)
→ Verwaltung von DBs unabhängig, Speicherung in **MASTER\Sysxlogins** System Tabelle, neue Server Roles können nicht hinzugefügt werden
Login Account Fixed Server Role zuweisen: über **EM** oder **sp_addsrvrolemember**, nur Mitglieder einer Fixed Server Role können auch logins hinzufügen
Syntax: **sp_addsrvrolemember 'login'**, 'role', login z.B. MOONSHEA\Paul
Fixed Server Roles: können nicht hinzugefügt/gelöscht/geändert werden, jedes Mitglied der Role kann weitere Mitglied Logins hinzufügen, NT user oder group kann Role zugewiesen werden auch wenn diese noch kein login besitzen, login wird automatisch mit **sp_addsrvrolemember** hinzugefügt, **sp_addsrvrolemember** kann nicht während benutzerdefinierter Transaktion ausgeführt werden, **sp_dropsrvrolemember** um Mitglied von Fixed Server Role zu löschen

---

9 Fixed DB Roles: **db_owner** (schließt alle Aktivitäten aller DB roles ein, hat erweiterte Wartungs und Konfigurationsvollmacht), **db_accessadmin** (Zugriffsverwaltung NT user/groups und SQLuser), **db_datareader** (kann alle Tabellendaten lesen), **db_datawriter** (... lesen/ändern/löschen), **db_ddladmin** (Objekte zufügen/verändern/löschen), **db_securityadmin** (Rolemanagement, Verwaltung von Statement und Objekt Berechtigungen), **db_backupoperator**, **db_denydatareader** (kann Schema ändern aber keine Daten lesen), **db_denydatawriter** (kann keine Daten ändern)
Public Role: jeder DB Benutzer gehört dazu, default permissions, kein manuelles Management von Usern/Gruppen/Roles da automatisch, jede DB enthält public role (auch System DBs), kann nicht gelöscht werden
Fixed DB Roles hinzufügen: über **EM** oder **sp_addrolemember**, nur **db_owner** role oder **role owner** kann hinzufügen
Syntax: **sp_addrolemember 'role', 'security_account'**, security_account ist z.B. db_datareader

→ Fixed DB Roles können nicht gelöscht/hinzugefügt/modifiziert werden, jedes Mitglied der Role kann neue Mitglieder bestimmen, **sp_droprolemember** um security_account aus role zu löschen

---

Benutzerdefinierte DB Roles: Gruppierung für gemeinsame Berechtigungen, wenn keine passende NT Gruppe besteht, wenn ich keine Berechtigungen besitze um NT User accounts zu administrieren, und bei Mixed Mode authentication
Benutzerdefinierte DB Role hinzufügen: über **EM** oder **sp_addrole** (Eintrag in **sysuser** Tabelle der DB), nur **db_securityadmin** und **db_owner** können **sp_addrole** ausführen
Syntax: **sp_addrole 'role', 'owner'**, owner ist User oder Role der BD
SQL Server Roles: können als Mitglied einer anderen SQL Server Role fungieren, rekursive Roles funktionieren nicht, auch nicht verwegt rekursiv, Role Verschachtelung mindert die Systemleistung
Weitere Prozeduren: sp_droprole (Server Role aus DB droppen), **sp_droprolemember** (Security_account aus Server Role entfernen)

KAPITEL12 – Permissions and Security Planning

Logins: Zugriff auf Server
Users (individ. User oder NT Group) and Roles: Zugriff auf DBs
Berechtigungen: damit User Objekte erstellen oder auf sie zugreifen darf, high-level user dürfen Objekte selber entwickeln
3 Berechtigungstypen: **statement** (CREATE DATABASE/DEFAULT/PROCEDURE/RULE/TABLE/VIEW, BACKUP DATABASE/LOG), **object** (SELECT, INSERT, DELETE, UPDATE, REFERENCES ~ in **EM** DRI, EXECUTE), **implied** (Fixed role, Object Owner)
Statement Berechtigungen: DBs oder DB-Posten erstellen, nur **sysadmin**, **db_owner** oder **db_securityadmin** dürfen statement permissions erteilen
Object Permissions: Datenarbeit und Prozedur-Ausführung (benutzt man ein **WHERE** clause, braucht man UPDATE und SELECT Berechtigung)
REFERENCE Berechtigung: für Tabellen, wenn Daten mit einem **FOREIGN KEY** abgeändert werden, muß SQL Server diese Daten validieren (hat Benutzer keine SELECT Berechtigung braucht er REFERENCE, in **EM** bezieht sich REFERENCE auf **DRI** (**Declarative Referential Integrity**)
Implied Permissions: synonym predefined oder implicit Permissions, erweiterte Funktionalität für (1) Fixed Roles und (2) DB owner
Fixed Role Permission: administrative Berechtigungen
Object Owner Permission: alle Aktivitäten von Objekten im Eigentum

---

3 Permission States: **GRANT** (kann von Role Membership überschrieben werden, granted permissions sind kumulativ), **DENY** (kann nicht von Role Membership überschrieben werden, daher auf jeden Fall Zugriffsverweigerung), **REVOKE** (es gab niemals GRANT oder DENY)
Permissions: Speicherung in **sysprotects** System Tabelle bei GRANT oder DENY, werden nur für eine DB ausgegeben, das Recht Berechtigungen auszusprechen besitzen nur **sysadmin**, **db_owner** und **db_securityadmin** roles und object owner, **CREATE DATABASE** Berechtigung kann nur user und Roles in der **MASTER** DB eingeräumt werden
Berechtigungen erteilen: über **EM** oder **GRANT** statement
Statement Permission Syntax: **GRANT {ALL | statement [,…n]} TO security_account [,…n]**
Object Permission Syntax: **GRANT {ALL [PRIVILEGES] | permission[,…n]}**
{
[(column[,…n])] ON {table |view}
| ON {table | view}[(column[,…n])]
| ON {stored_procedure | extended_procedure}
}
TO security_account [,…n] → **bis hierhin auch so mit DENY !!!**
[WITH GRANT OPTION]
[AS {group | role}] → **bis hierhin ohne with grant option mit REVOKE**

→ **ALL** heißt alle Statement Berechtigungen, bei Objektberechtigungen heißt **ALL** alle Objektberechtigungen die für ein Objekt möglich sind, nur **system administrator** und **database owner** können **ALL** benutzen, außerdem müssen WINNT usernamen in eckigen Klammern angegeben werden, z.B. [MOONSHEA\Paul]
Wenn REVOKEs keine REVOKEs sind: kommt auf das System an wo die REVOKEs ausgesprochen werden (für user oder roles), REVOKEs können über kompliziertes Berechtigungs-System Berechtigungen verweigern oder bestätigen !!!

Sicherheitsplanung: Ziele: Alle Aktivitäten und Items zur Kontrolle auflisten, Identifizierung von Gruppen und individuellen Usern, Cross-Referenz zwischen Usern und Aktivitäten
5 Considerations:

1. **Default Logins**: **sa** (es sollte eigene Logins für sysadmins geben, pwd auf jeden Fall ändern, da default ohne), **BUILTIN\Administrators** (default Mitglied von der sysadmin role, kann als login oder auch aus der role entfernt werden, kann dann auch wieder hinzugefügt werden, Entfernung kann auch über das Wegnehmen der **Domain Admins Global Group** aus der **Local Administrators Group** gemacht werden)
2. **Public Role Berechtigungen**: default public role hat "no permissions"
3. **Guest User Account**: berechtigt zum login ohne Account, neue DBs haben kein guest
4. **Mapping von Logins zu User Accounts und Roles**: gibt es NT Group, dann für diese einen User Account anfertigen, bei Usern mit gleichen Tasks Roles erstellen, Fixed Server und DB Roles benutzen !!!
5. **dbo**: dbo ist Objekt-Eigentümer wenn von sysadmin geschaffen, immer spezifizieren weil sonst user name benutzt wird (kompliziert bei nachträglichen Beziehungen, da Objekt-Eigentümer bekannt sein muß), kann geändert werden über **sp_changeobjectowner**

Syntax: **sp_changeobjectowner [@objname =] 'object' ,[@newowner =] 'owner'**, nur Mitglieder der **db_owner**, **db_ddladmin** und **securityadmin** können ändern

---

Verwaltung von Applikationssicherheit: zusätzliche Sicherheit auf Applikationsebene
Über Views und Prozeduren: Sicherheit für SQL-Objekte für Applikationen, Berechtigungen erteilen
Views: Tabellenspalten können vor Usern versteckt werden (TIP: View Berechtigungen sond einfacher als Spalten Berechtigungen), Berechtigungen also an Views und nicht an Tabellen vergeben
Prozeduren: Berechtigung an Prozeduren und nicht an Tabellen
Application Roles Unterschiede zu anderen Roles: keine Mitglieder (Applikationen aktivieren Application Roles), benötigen ein Pwd um zu funktionieren, zum Schutz des Role Pwd bspw. Verschlüsselung oder über erweiterte Prozedur Lagerung des Pwds auf dem Server, Berechtigungen überschreiben User Berechtigungen in der DB
Application Roles einrichten: über **EM** oder **sp_addapprole**, nur **db_owner**, **db_securityadmin**, **sysadmin role** kann ausführen
Syntax: **sp_addapprole [@rolename =] 'role', [@password =] 'password'**, schafft security account in sysusers Tabelle der DB, Pwd wird verschlüsselt gespeichert
Application Roles aktivieren: nach Client-Verbindung mit login account, Client Applikation muß **sp_setapprole** ausführen (kann nur über direkte Transact-SQL statement ausgeführt werden, nicht über andere Prozedur oder benutzerdefinierte Transaktion)
Syntax: **sp_setapprole [@rolename =] 'name' , [@password =] {Encrypt N 'password'} | 'password' [,[@encrypt =] 'encrypt_style']**, Applikation muß Pwd bereitstellen, application roles gelten nur für eine DB, Deaktivierung der Role geht erst bei Server-Disconnect vonstatten
Weitere Prozeduren: **sp_dropapprole**, **sp_approlepassword** (Pwd ändern)

KAPITEL13 – Automating Administrative Tasks

Automatisierungsgründe: für Routine-Wartungsaufgaben, potentielle Probleme erkennen und darauf reagieren
Reguläre Scheduling Aufgaben: Backup, Datentransfer, Indexwartung

Problemreaktion: auf bestimmte Server-Errors reagieren (z.B. **9002** → Transaktions Log voll → back up and truncate), Monitoring von Performance Conditions (z.B. durch User erzeugte Daten-Locks)
Komponenten der SQL Server Automated Administration: Server Services, Agent (früher: **SQL Server Executive**), NT Event Log Service
Agent Service: wird bei Start beim Event Log Service (Notifier über Application Log) registriert und mit Server Service verbunden (Aktionsinitiator)
Aktionen: jobs, alerts, definiert in MSDB DB
Einträge im Event Log wenn: Fehler mit Sicherheitslevel zwischen **19** und **25**, definierte Fehlerbenachrichtigungen über **sp_addmessage** oder **sp_altermessage**, bei Ausführung **RAISEERROR WITH LOG** statement, Ausführung **xp_logevent**
Win9X: anstatt Event Log Service wird **SQL Server Profiler** benutzt
Jobs und Alerts: separat definiert und unabhängig, **Jobs** (Task mit einem oder mehr Steps, scheduled oder Alert-Antwort), **Alerts** (potentielle Probleme, von Event Log informiert oder Performance Benachrichtigung vom Server), Jobs und Alerts können eine reziproke Beziehung haben
Vorbereitungen: **Benachrichtigung über Email oder Pager** (Server Agent Mail Profile konfigurieren, **MAPI-1** kompatiblen Email Client, **Agent Mail Session** wird bei Start des Agent Service mitgestartet), **Agent Service** (automatischer Startup), **Server Agent User Account** (lokales Systemaccount – nur lokal – oder Domain User – email, Netzwerk, Replikation, multiserver jobs),
Agent Mail Profile: z.B. mit Outlook einrichten, bei Verwendung Exchange Server muß Profil mit Domain User Account ausgestattet sein, Pager Benachrichtigungen benötigen 3rd Party Software oder Hardware zur Konvertierung
Profil mit SQL Mail sharen: über **2** separate Mail Sessions: **SQL Mail** (vom Server benutzt bei Verwendung **xp_sendmail** beim Senden und **sp_processmail** beim Empfangen), **Agent Service Mail Session** (exclusiv), bei Benutzung der gleichen Domain User Accounts gibt es nur **1** Mailbox für beide
2 Möglichkeiten Separate Profile einrichten: separate domain user accounts oder gleiches account und multiple mail profiles
Workgroup Postoffice einrichten: Admin einloggen → Systemsteuerung – Add/remove Programs → in Setup Windows Messaging/Microsoft Mail checken → Systemsteuerung Refresh → Doppelklick Postoffice → neue User über Postoffice Manager anlegen

Benachrichtigungsoperatoren einrichten: in Application Log, Jobdelete, Pager, Email, **net send** command
Neue Operatoren einrichten: **EM** oder **sp_add_operator**, in **MSDB_sysoperators** gespeichert
Guidelines: email aliases für Gruppenbenachrichtigung möglich, immer testen, für jeden Operator **on-duty schedule** einrichten (sonst Konflikte möglich), Nachrichten an Netzwerkoperatoren mit net send (nicht auf Win9X)
Troubleshooting wenn keine Notifications eingehen: Operator empfangsbereit?, Log nach Datum und Zeit durchsuchen!, andere Methoden außerhalb SQL Server benutzen um Sendemöglichkeit überhaupt festzustellen
Neue Jobs einrichten: **EM** oder **sp_add_job**, **Create Job Wizard**, in **MSDB_sysjobs** gespeichert (wegen Performance immer im Cache)
Guidelines: enable Job (default) → disabled kann nur bei Alert Fire oder manuell in EM gestartet werden, owner spezifizieren (default login account), Jobdefinition lokal oder multiple remote servers, Job Kategorien einrichten
Job Steps definieren: über **EM** oder **sp_add_jobstep**, gespeichert in **MSDB_sysjobsteps**
Job Steps: SQL Befehle, Repliaktionsaufgaben, OS Befehle, Active Scripts → jeder Step kann nur von einem Typ sein, verschiedene Typen können im Job kombiniert werden

1. **SQL Befehle**: Variablen und Parameter im Step angeben, **Result Set** des Steps kann an **Output File** ausgegeben werden (Output kann aber nicht wieder als Input verwendet werden), User Context wird über **SETUSER** spezifiziert (wenn Jobowner (NT oder SQL

account) nicht Mitglied in sysadmin role ist, oder ist doch Mitglied aber Job läuft unter anderem Kontext

2. **OS Befehle**: .EXE, .BAT, .CMD, .COM, process exit code für Erfolgsbestätigung, vollen Pfad zur Datei bestätigen
3. **Active Scripting Languages**: Visual Basic, VB Script, Javascript

Berechtigungen für OS Befehl oder Active Scripting Step Ausführung: default alle Jobeigentümer, sysadmin role führt im **Agent** account aus, andere im **AgentCmdExec** NT user account, unter **SQL Server Agent Properties** kann die Ausführung allerdings auf sysadminrole beschränkt werden durch Option Flag setzen
Action Flow Logic für Steps bestimmen: default bei Erfolg kommt neuer Step, bei failure stoppt step, es können aber auch steps eingerichtet werden, die bei Erfolg oder failure in Kraft treten, retry bei failure und retry intervals können auch bestimmt werden
Scheduling Jobs: über **EM** oder **sp_add_jobschedule**, gespeichert in **MSDB_sysjobschedules**
Jobs können ausgeführt werden wenn: job und schedule enabled, Zeitpunkt oder regelmäßig wieder, können automatisch gestartet werden wenn Agent gestartet wird oder CPU idle ist
Idle CPU Condition: in Agent Properties, default wenn durchschnittl. CPU Gebrauch unter **10%** für **600** Sekunden
Multiple Schedules: ein job kann mehrmals getimed werden
Job History: Agent speichert Job und Steps in **MSDB_sysjobhistory**, Größe der History Tabelle kann angegeben werden über EM-Agent Properties (Größe in Tabellenreihen, kann unbeschränkt bleiben oder durch MSDB autogrowth limit beschränkt werden, default **1000** rows, job **100** rows, History wird nach **FIFO** aufgeräumt) → IMMER HIER NACH JOBFEHLERN ZUERST SUCHEN!
Job History Record: Date und Time, fail or success, notification method und Operator, Step-Dauer, Fehler oder Benachrichtigung
Bei Benachrichtigungsverzögerung an Operator: liegt evtl bei failure des job (step) an retry!!!
SQL-Mail: unter Support Services in EM → Server und Agent benutzen unterschiedliche Profile

Alerts: antworten auf benutzerdefinierte oder SQL Server Fehler, die ins NT application log geschrieben werden
RAISEERROR statement: von DB ausgeführt wenn benutzerdefinierte Mitteilungen erfolgen sollen, nicht vom SQL Server !!!
Alerts bei SQL Server Error Numbers: über **EM**, **Create Alert Wizard** oder **sp_add_alert**, Alert Definition gespeichert in **MSDB_sysalerts**, ständig im cache
Alert guidelines: Alerts des SQL System oder benutzerdefinerte Fehler Nummern werden in **master_sysmessages** gespeichert, error number kann mehrere alerts haben, alert für eine DB oder alle spezifiziert, error message sollte DB Namen enthalten
Fehler 18456: logon failure of user
Severity Level: spezifische Fehlerkategorie, **0-10** informelle Benachrichtigung, **11-16** Benutzerfehler, **17** für insufficient ressources, **18** Non-fatal error, **19** nonconfigurable resource error WRITTEN TO LOG, **20-25** fatal errors WRITTEN TO LOG
Fatal errors: Code wird terminiert und Verbindung mit User unterbrochen
Error Message einrichten: über **EM** oder **sp_addmessage**, Fehlernummer muß größer **50000** sein (unter **50000** vordefiniert), Parameter können verwendet werden (um DB oder Username anzuzeigen), Mitteilungen immer in Installationssprache oder selbstdefiniert, error message müssen in NT log um einen Alert darauf zu erzeugen
Event Forwarding: ungehandelte error messages können an andere Server weitergeleitet werden, kann für aufsteigende severity levels angegeben werden, bspw. ab 19, so müssen in Multiserver Umgebungen alerts nur einmal definiert werden (am besten auf server mit wenig traffic)

Auf Performance Condition alerts antworten: definiert über Objekte und Counter in NT Performance Monitor, alert bei **Wertübersteigung**, **equal** oder **darunterfallen**, → **Performance Monitor** muß nicht laufen !
Fail-Safe-Operator: auf Alert antworten wenn pager Benachrichtigung an Operatoren nicht funktionieren (es liegt am pager oder Agent kann Systemtabelle in MSDB nicht einsehen), immer gecached, es gibt nur einen, Operatoren können wenn sie als Fail-Safe Operator bestimmt wurden nicht gelöscht werden, nur wenn zuerst der Fail-Safe Operator geändert wird.

Troubleshooting Automated Administration:
Checklist: Agent started → job schedule alert operator enabled → SQLAgentCmdExec account konfiguriert → error logs → job history → mail client

1. **Agent started**: sollte automatisch starten, SQL Server Agent Monitor macht Selbstcheck falls Agent unerwartet stoppt und versucht ihn neuzustarten, Aktivierung des Monitors über **EM-Server Properties-advanced** oder **xp_sqlagent_monitor**
2. **...Enabled**: wird nicht in Agent Fehlerlog oder Job History aufgenommen wenn nicht gestartet
3. **AgentCmdExec**: muß da sein wenn Eigentümer des Jobs – nicht sysadmin role – Active Scripting oder OS command ausführen will (Passwortänderung unter NT aber nicht Agent Properties, oder flag für nur sysadmins gesetzt, account gelöscht oder falsch installiert)
4. **Error Logs**: **NT Application Log** (maximum size!, overwrite frequently kann Reaktion verhindern), **Agent Error Log** (execution trace messages Option sollte nicht eingeschaltet werden wegen Größenzuwachs, bis **9** Einträge in **c:\mssql7\log** möglich), **Server Error Log** (Zeitvergleich mit anderen Logs vornehmen um Reaktionskette festzustellen)
5. **Job History**: in MSDB, maximum Größe job history herunterschrauben da sonst volle sysjobhistory Tabelle kann Probleme bereiten, also MSDB vergrößern
6. **Mail Client**: in Mailprofil einloggen als SQL Server Agent Domain User Account und Tests durchführen

Endless Loops: da Agent gleizeitig abhängig von Serverevents aber diese auch monitored, bei essentiellen globalen Ressourcen, Indikator z.B. wenn sich NT Log füllt mit gleichem Fehler, CPU Benutzung durchgängig hoch, Anzahl der alert responses hoch), erzeugt Alert-backlog
Backlog reinigen: über **NT Event Viewer** löschen (dann werden alle Einträge gelöscht) oder **Response Delay** nutzen um Alerts zu reduzieren oder Definition als **Non-Alert-Generating** über Modifikation der Registry (nur Notmöglichkeit)

Mulitserver Environment:
Multiserver Administration Modell: **Master Server** (**MSX**) und einer oder mehrer **Target Server** (**TSX**, können aber müssen nicht registriert werden), zur Grupierung logischer Geschäftseinheiten, zentrales Management, benötigt auf allen Beteiligten SQL Version 7
Master Server definieren: hat eingelistete Targetserver (mindestens einer, gespeicher in **systargetservers** auf Master Server), sollte auf NT laufen, über **EM** oder **Make MSX Wizard** (richtet MSXOperator auf master und jedem target ein), repräsentiert primary department, dieser sollte auch event forwarding server sein
Target Server definieren: über **EM** oder **sp_msx_enlist** oder **Make Target Server Wizard**, gespeichert in **MSDB_systargetservers**, TSX gehören nur zu einem MSX, befinden sich in gleicher oder vertrauter Domäne wie MSX, können kein Mitgleid eines anderen MSX sein bis Defekt
Automating Jobs in ME: jobs auf master einrichten um auf target auszuführen
Prozess: MSX postet jobs in **MSDB_sysdownloadlist** → TSX überprüfen regelmäßig auf Neuigkeiten und downloaden wenn so → TSX uploads Outcome Status von erfolgten multiserver jobs
Multiserver Job Definitionen ändern: nur auf MSX, EM postet Instruktionen an download list

Job History: zusätzlich zur normalen Job History der einzelnen TSX speichert der MSX alle zentral bei sich

KAPITEL14 – Monitoring und Maintaining SQL Server

Gründe: **Poor Performance** (zu lange Response Times ~ Perceived Time),**Throughput of all processing for all users** (Durchsatz bei Queries etc.), **Consistency of Data** (Datenbeschädigung, check über DBCC)
6 Ursachenkategorien: **Hardware** (Prozessor, Disk I/O Disks und Controller, RAM), **OS** (andere Services, paging files, RAID), **Network** (Bandbreite, Transferrate), **SQL Server** (Konfiguration, Locking, Logging, concurrent activities wie Wartung, backup n restore, DBCC, indexing), **DB Applicaction** (logisches und physisches Design, Transaktionskontrolle, Konflikte, Queries), **Client Application** (Transaktionskontrolle, client response to locking conflicts, cursor)

Performance Baseline: Monitoring over time zu Vergleichszwecken
Query Governor konfigurieren: um unwirtschaftliche Queries zu vermeiden, funktioniert über geschätzte Dauer in Sekunden, Query Governor hat Beschränkungswerte, option **0** läßt alle Abfragen ohne weitere Nachfragen laufen, über **EM** oder **sp_configure** kann derverweite Änderung vorgenommen werden, bei individuellen Verbindungen kann über **SET_QUERY_GOVERNOR_COST_LIMIT** individuelle Settings eingestellt werden

Monitoring Werkzeuge: SQL Server Performance Monitor, Transact-SQL statements, SQL Server Profiler, SQL Server Query Analyzer, Current Activity in SQL Server EM
Examinationsvorgang: erst Systemebene (Event Viewer und Performance Monitor) vor Client- oder Queryebene, dann SQL Server Activity (locking und contention, gleichzeitige Verbindungen und tempdb Benutzung über Prozeduren Profiler EM und SQL statements), dann Konsistenzüberprüfung (DBCC), dann Query Performance (index, CPU, I/O über Profiler, Query Analyzer und Index Query Wiz)
Benutzung des Event Viewer: Fehlerbenachrichtigungen können dem Typ nach gefiltert werden
SQL Server Performance Monitor: für Aktivitätsinformationen und Statistiken, eigentlich NT Performance Monitor aber erweitert um vordefiniertes Set von Countern in der Datei **sqlctrs.pmc**, **16** Objekte mit mehr als **100** Countern, counter können auch selber definiert werden in **.pmc** Dateien, PM ist aber nicht unter 9X verfügbar oder von hier auf NT benutzbar!
Sicherheit: bei NT authentication ist Benutzung des PM **sysadmin role** vorbehalten
Counter: Seite 400
Benutzerdefinierte Counter: bis **10** möglich, über **SQLServer:User Settable** object, zu beobachtende ,Werte können über **sp_user_counter1** zugewiesenwerden, zu beobachtende Ereignisse können SQL statements oder irgendeine Operation, die einen Wert zurückgibt sein, z.B. Prozedur
CTRL-H: spezielles Counter Objekt in Performance Monitor highlighten

Monitoring mit SQL Prozeduren: **sp_who** (derzeitige SQL Server user und Prozesse), **sp_lock** (aktive locks), **sp_spaceused** (HDD von Tabelle oder DB benutzt), **sp_helpdb** (DBs und ihre Objekte), **sp_monitor** (SQL Server Statistiken), **sp_helpindex** (Index einer Tabelle)
Monitoring mit SQL Funktionen: starten alle mit @@, **@@CONNECTIONS** (Verbindungen seit Serverstart), **@@CPU_BUSY** (CPU Arbeit in Millisekunden seit Serverstart), **@@IO_BUSY** (IO in Millisekunden seit Start), **@@IDLE** (idle in Millisec seit Start), **@@TOTAL_ERRORS** (Anzahl disk/write errors seit Start), **@@PACKET_ERRORS** (Netzwerkpaketfehler bei SQL Verbindungen seit Start)
SQL SET-statements: **SET STATISTICS TO IO** (info über disk Aktivität), **SET STATISTICS TIME** (Millisekunden für parse, compile und execute statements), **SET SHOWPLAN_TEXT** (keine Ausführung sondern detaillierte Info über statement Ausführung)

DBCC statements: **SQLPERF** (Transaktionslogspace benutzt), **OPENTRAN** (älteste aktive Transaktion in einer DB), **SHOW_STATISTICS** (index selectivity), **SHOW_CONTIG** (Datenfragmente und Tabellenfragmente eines Index), **CHECKDB** (Allokation und Integrität der DB Objekte), **CHECKFILEGROUP** (... Tabellen in filegroup), **CHECKALLOC** (Allokation und Benutzung von Pages in DB), **CHECKTABLE** (Integrität von Daten, Index... in Tabelle)
Trace Flags: mit **DBCC TRACEON** setzen und mit **DBCC TRACEOFF** disablen, S. 406, bei den meisten sollte man SQL Server am command prompt starten oder an error log über traceflag **3605** weitergeben, sie werden von Microsoft aber nicht supportet

SQL Server Profiler: Daten können monitored werden oder in Tabellen, Dateien oder SQL script gecaptured werden, traces können public (alle diesen Computer Benutzer) oder privat sein, Eventfilterung und Grouping möglich, real-time results, gut um lange queries von usern oder login failures aufzudecken
Warum Trace Informationen speichern: Aktivitäten Replay, Analyse und Filtering, **load file** für Index Tuning Wiz, debug applix
Index Tuning Wizard: einrichten eines optimalen Index-Set und Statistiken für eine DB ohne fachliches Hintergrundwissen, über EM oder Profiler, für Empfehlung benötigt der Wiz das **workload** (besteht aus SQLscript oder SQL Server Profiler Trace)
Query History: Trace der letzten **100** Events im SQL Server, welche Prozeduren des Profilers benutzen, kein erwähnenswerter Performance Verlust zu enablen, Speicherung in **blackbox.trc** in **c:\mssql7\log** bei Ausnahmefall mit severity **17** und höher
Query History aktivieren: **execute xp_trace_setqueryhistory (0|1)**, **1** enabled
Query History in Trace File schreiben: **execute xp_trace_flushqueryhistory 'filename'**
Datei mit Profiler anschauen: blackbox.trc oder eigengespeicherte mit **xp_trace_flushqueryhistory**, SQL Server muß dabei nicht laufen!!!

Query Analyzer: grafische Darstellung über **show execution plan**, Icons repräsentieren steps, Mauszeiger darauf zeigt Statistiken der einzelnen Steps
Index Analysis Tool: Indexes für bestimmte queries empfehlen
Current Activity in EM: ***Management-Current Activity*** → Info über aktuelle Prozesse (**1** Prozess pro User Verbindungen, Systemprozesse: locks), Nachrichten an Benutzer schicken, Prozesse terminieren, current Activity wird nicht automatisch aktualisiert!
Locks: um simultane interferierende Transaktionen zu verhindern, locks für Tabellen und data pages, **2** Views: Locks/ProcessID, Locks/Object
Exclusive Locks: bei **UPDATE**; **INSERT**; **DELETE**; halten immer bis zum Ende der Transaktion an, weitere User kommen dann nicht mehr an Daten ran
Shared Locks: für Leseoperationen, Benutzer können lesen aber nicht mehr ändern, bestehen für die Dauer der Leseoperation, mit **HOLDLOCK** wird die Dauer auf das Ende der Transaktion dennoch erhalten
Blocking: Transaktionen, die auf ein Objektlock warten, heißen blocked und haben den Status **WAIT**. Diese Transaktionen haben kein time-out es sei denn sie verwenden **LOCK_TIMEOUT**, blocking hat nichts mit deadlock zu tun
Deadlock: wenn **2** User locks auf separate Objekte haben und vergebens gegenseitig auf die Freigabe des Objektes warten, SQL Server bemerkt das und erklärt eine der beiden Transaktionen als Opfer und macht dessen seitherige Transaktion rückgängig, dieser Benutzer erhält Fehlermeldung **1205**
Erzeugter Lock: wenn Transaktionen mit BEGIN TRAN gestartet, aber kein COMMIT TRAN oder ROLLBACK TRAN zur Komplettierung der Transaktion existiert

Maintaining SQL Server Kernaufgaben: aktualisieren von Infos über Datenoptimierung, Datenintegrität überprüfen, Backups, Berichte erstellen und Wartungshistorie der DB

1. **Datenoptimierung Update**: Daten und Index Pages reorganisieren (Prozentsatz des fillfactor spezifizieren um Performance zu steigern), UPDATE STATISTICS (auf modifizierten Tabellen, erzeugt key-value update), DBCC SHRINKDATABASE (um in Tabellen ungenutzten Platz zu entfernen)

2. **Datenintegrität überprüfen**: **DBCC CHECKALLOC** (checkt Allokation von Daten- und Index Pages), **DBCC CHECKTABLE** (in Tabelle), **DBCC CHECKDB** (überprüft Objekte in DB, enthält die Befehle DBCC CHECKALLDOC und DBCC CHECKTABLE, am sichersten mit der Repair-Option starten, während Befehl läuft kann keine Tabellenoperation ausgeführt werden), wenn DBCC einen Fehler findet, kann dieser automatisch repariert werden.
3. **Backups**: q.e.d
4. **Maintenance History**: als Dokumentation, als Report in Datei, in History Tabellen, als email an Operator

Automatic Maintenance: **Maintenance Plan** mit **Database Maintenance Wizard** oder **Sqlmaint Utility** erstellen
Database Maintenance Wizard: kann Kern-Wartungsaufgaben erstellen, erstellt Plan für eine, mehr oder alle DBs
Sqlmaint Utility: Befehlszeilenprogramm für Funktionen ähnlich DB Maintenance Wiz

KAPITEL15 – Introducing Replication

Warum Distributed Data: um Abfragen effizienter zu machen, Daten allen zugänglich machen, Daten näher an User bringen, Sites operieren autonom und unabhängig, **OLTP** von leseintensiven Applikationen wie z.B. Data Warehouse und Data Marts separieren, Konflikte reduzieren
Distributed Data Environment: viele Kopien mit der gleichen Information auf multiplen Servern, zum Teil auch unterscheidliche DB Management Systeme aus früheren Unternehmensentstehungszeiten
2 **prinzipielle Strategien für verteilte Daten**: **Distributed Transactions** und **Replication**
Was ist eine Transaktion: Serie von Datenänderungen, die vorzugsweise vervollständigt werden muß
Unterschiede zwischen DT und Replication:

1. **DT**: Daten auf beiden Servern immer konsistent und synchronisiert (Modell nur dann verwenden), basiert auf **two-phase-commit protocol**, Methode ist weniger anpassbar als Replikation, Transaktionsfehler werden an alle Beteiligten weitergegeben, wird verwendet bei Applikationen, die simultane Updates an mehreren DBs vornehmen
2. **Replikation**: kürzliche Datenkopien werden dupliziert und von Source DB an Ziel DB abgegeben, autonome Sites möglich (d.h. Server müssen nicht ständig online sein), DBs sind auf grossen Servern und Einzelbenutzercomputern

Entscheidungsfaktoren für Strategiewahl: **Latency** (Wartezeit, Delay bei Updates von Datensets, bei DT nahezu Null), **Site Autonomy** (permanenter Verbindungsstatus online oder nicht), **Transactional Consistency** (Replikationen führen nicht immer komplette Updates ducrh und gewähren somit keine absolute Konsistenz), **Database Update Conflicts**, **Security**, **bestehende Datenquellen**, **Performance**, **Administration**
4 **Methoden der Datendistribution**: **Distributed Transactions** (alle Sites haben alle Daten zur gleichen Zeit, benutzt **MS DTC**), **Transactional Replication** (nur veränderte Daten werden übergeben, es gibt nur einen Datenänderungsort, keine Konflikte), **Snapshot Replication** (Source Server koordiniert über eigenen Mirror periodisch oder bei Bedarf), **Merge Replication** (multiple unabhängige Sites replizieren periodisch über Source Server)
Video Keypoints: Replikationen starten alle typischerweise mit SNAPSHOT Replication. Distribution DB speichert bei allen Modellen nur Statusinfos und keine Daten, außer bei TRANSACTIONAL Replication

Publisher-Subscriber-Metaphor: **Publisher** (Datensender), **Subscriber** (Empfänger, hält Kopie, kann für andere wiederum als Publisher fungieren), Replikationsmodell, beinhaltet auch Filterung
SQL-Server: kann als Publisher (**SOURCE DB**), **Distributor** (empfängt und speichert alle Änderungen, forwarded an Subscriber, Vermittler), Subscriber oder eine beliebige Kombination der drei sein

Distributor: **DISTRIBUTION DB** und **DISTRIBUTION WORKING FOLDER** werden als Vermittlungsort (Daten + Statusinfos) angelegt, default Distributionsordnerlokation **c:\mssql7\repldata** (änderbar oder andere Ordner hinzufügbar), kann Änderungen an Subscriber senden oder diese können Daten auch unangemeldet abholen, befindet sich der Distributor auf einem anderen Computer, braucht dieser eine komplette eigene lizensierte Serverinstallation
Wichtigstes Replikationskonzept: **jedes replizierte Datenelement hat nur einen Publisher** → es gibt kein multiple-master Modell!
Publications: für Replikation markierte Daten, Sammlung von Articles, Subscriber subscriben sich für Publications (nicht für Articles, geht aus Rückwärtskompatibilität aber geht nicht über **EM**), enthält alle notwendigen Daten für Applikation oder Operation, mehrere können für DB eingerichtet werden, kann aber nicht für mehrere DBs gültig sein, immer nur eine
Articles: grundlegendes einzelnes zu replizierendes Datenelement einer Replikation, z.B. eine Tabelle, spezielle gefilterte Zeilen oder Spalten, Prozedurdefinitionen, das Ausführen einer Prozedur (zur Verminderung der Datenübertragung bei TRANSACTIONAL R.)
Datenfilterung: Tabellensubset kann gepublished werden, **vertikale** oder **horizontale** Filterung, jede gefilterte Instanz ist ein separater Artikel, benutzt zur Konfliktvermeidung
Vertikale Filterung: Spalten, ähnlich **SELECT** statement, geht bei MERGE R. nicht, **z.B. ohne Gehaltsspalte**
Horizontale Filterung: Zeilen, ähnlich **WHERE** clause in **SELECT**, **z.B. Bestellungen nach Region**
Weitere Wege um Datensubsets zu erzeugen: direkt separate Tabellen anlegen (**z.B. für jede brach**), sogenannte **PARTITIONIERUNG** (bei Spalten eher unüblich), Administration muß dann aber mit mehr Tabellen umgehen!
2 **Subscriptions Arten**: PUSH Subscriptions, PULL Subscriptions
PUSH Subscriptions: beim Publisher definiert, mehrere Subscriber können auf einmal für jede Publikation zugelassen werden, Distributor propagiert ohne request des Subscribers (höhere Sicherheit, real-time updates ~ schneller, wenn Distributor genug Prozessorkraft hat), Replikationsagenten laufen auf Distributor oder Publisher
PULL Subscriptions: Subscriber initiiert, Publikation muß für pulls enabled sein, nur SQL Server Subscriber unterstützen PULL, admin oder DBowner entscheidet wann und was, Agenten laufen auf Subscriber
2 PULL Arten: **Standard** (beim Publisher registriert), **Anonymous** (beim Subscriber registriert, anonym ohne Informationsangabe, ideal für Internet z.B. über **FTP**)
TIP: Erstellung und Administration von Subscriptions kann physisch über **EM** natürlich auch remote geschehen, einfach entsprechenden Server auswählen

3 **Replikationstypen**: **snapshot**, **transactional**, **merge**, unterschiedlich in der Verwendung von Agenten und Ressourcen, mehrere Replikationstypen können mit der gleichen DB benutzt werden
SNAPSHOT: periodischer bulk Transfer einer vollständigen Publikation an Subscriber, am leichtesten zu implementieren und warten, hohe Wartezeiten (periodisch), hohe Site autonomy, benötigt keine **primary keys**, nicht geeignet für große Publikationen, wenig Prozessoroverhead da kein kontinuierliches Monitoring
Snapshot Repliaktionsprozeß: Snapshot Agent liest **PublikationsDB** und generiert Snapshot Dateien im distribution working folder (auf dem Distributor, auch snapshot folder genannt), Statusinfo in Distribution DB, Distribution Agent (PUSH auf distributor, PULL auf Subscriber) holt Datensnapshots zum Subscriber
Distributions DB: nur Statusinformationen, keine Daten!

TRANSACTIONAL: inkrementelle Änderungen, minimale Latency (Sekunden), niedrigerer Grad an site autonomy, braucht **primary keys**, für jede Publikationsgröße geeignet, kontinuierliches Monitoring, nur komplett übermittelte Transaktionen werden repliziert und zwar nur in der gleichen Reihenfolge, die **SQL Desktop Edition** hat keine transactional publication, kann sich aber subscriben

Transactional Replikationsprozeß: startet wie Snapshot mit Snapshot Agent, Distribution Agent nimmt Dateien von Snapshot Agent und beginnt mit transactional replication (benutzt den Log Reader Agent, um Transaktionslog auf Publisher periodisch zu lesen, die gelesenen Infos schreibt er in Distribution DB), Distribution Agent verteilt dann Änderungen (Agent bei PUSH auf Distributor, bei PULL auf Subscriber)
Immdiate Update Subscribers Option: bei transactional und snapshot werden Daten nie auf Subscriber modifiziert, da **one-way-data-flow** (Änderungen würden nie berücksichtigt werden), Tabellen können aber für **two-way partitioniert** werden (d.h. Server ist für einen Teil Publisher und für anderen Teil Subscriber), in **v7** aber auch neue immediate update option: kombiniert TR und SR mit two-phase commit protocol (von **MS DTC** verwaltet), Subscriberänderung geht wenn Änderungen zur gleichen Zeit an Publisher weitergegeben werden können (benötigt ein dauernd verbundenes verläßliches Netzwerk)
Vorteile: der Benutzer auf Subscriber kann unterbrechungsfrei arbeiten, Änderungen kommen überall an
Benutzt man immediate Option mit snapshot: sollten publizierte Daten **partitioniert** werden, da veränderte Reihen bis Replikation nicht mehr modifiziert werden können (ist es doch so dann fail)

MERGE: autonome Veränderungen, hohe site autonomy, latency wie auch immer, Tabellen haben einzigartige Identifier Spalte, für alle Publikationsgrößen, Tabellentrigger markieren zu replizierende Reihen, keine absolute Konsistenz aber Einheitlichkeit im Endeffekt, kein vertikales filtering, geeignet für aufgeteilte business-Funktionen
Merge Replikationsprozeß: beginnt mit snapshot replication, dann merge agent (PUSH auf distributor, PULL auf subscriber, in Distributions DB nur Status Info und keine Daten) holt Daten von Publisher auf Subscriber, dann revers zurück und Konfliktbereinigung
Considerations: es gibt **3** Schemaänderungen in der PublikationsDB:

1. **Spaltenidentifier** und hat Eigenschaft **ROWGUIDCOL**, gibt es diese Spalte nicht wird sie hinzugefügt
2. **Systemtabellen** für data tracking, effizienter Synchronisation, conflict detection, resolution, reporting
3. **Triggers** in Tabellen auf Publisher und Subscriber für jede Reihe (optional Spalte), Trigger speichern Änderungen in merge system Tabellen

Triggerkonflikt: MR Triggers und Appliaktions Triggers können koexistieren und interferieren nicht
Konfliktbehebung: Priority-basiert, bei mehreren Änderungen Kontrolle über **lineage** (Abstammung), **Evaluation** (Auswertung) von alten/neuen Daten über **Priorities**, alle Sites haben am **Ende** die gleichen **Datenwerte**

4 Replikationsagenten: separate Prozesse auf Server Agent, Publications und Subscriptions haben assoziierte **named agents**, ein Agent-Job, der Replikationsagenten konfiguriert, Agent Jobs haben Kategorie **REPL <agentname>**

1. **SNAPSHOT AGENT**: Synchronisation vor Replikation (alle Typen), verantwortlich für Dateispeicherung, läuft auf Distributor, bei SR auch für Replikationsprozeß zuständig
2. **DISTRIBUTION AGENT**: bei SR und TR für Datenbewegung von Publikationen verantwortlich, Lokation abhängig ob PUSH oder PULL
3. **LOG READER AGENT**: kopiert in Transaction Log markierte Transaktionen (auf Publisher) in distribution DB, läuft auf Distributor und bewegt Daten vom distributor zum publisher
4. **MERGE AGENT**: fügt Daten aus div. Sites zusammen seit initial snapshot, **2** Bewegungsrichtungen, Lokation abhängig von PULL oder PUSH

Entwicklung: eigene Subscriber Applikationen für Dist. Oder Merge Agent über **SQL Distribution Control** oder **SQL Merge Control** (**ActiveX** controls bei **v7** mitgeliefert)

3 Physikalische Replikationsmodelle:

1. **Central Publisher/Distributor Model**: 1 oder 2 P/D, viele S, P und D können auf gleichem Server sein, P Server ist primary owner der Replikationsdaten, D speichert vor

Repliaktion Daten zwischen, replizierte Daten auf S sind read-only, nur SELECT von Admins gegeben auf replizierte Tabellen

2. **Central Subscriber/Multiple Publishers Model**: viele P/D, 1 S, zur Zentralisierung von Daten, alle Daten brauchen einen einzigartigen lokalen Eigentümer (um Überschreiben zu vermeiden, erreicht durch horizontale Filterung)
3. **Multiple Publishers/Multiple Subscribers Model**: spielen alle eine potentielle duale Rolle, ähnelt am ehesten fully distributed data processing, Konsistenz muß sorgfältig über horizontales Filtern und Partitionierung implementiert werden

Kombination von Modellen und Typen: Modell ist physikalische Implementierung des Replikationsdesign, Typ stellt Funktionsweisen dar, Modell und Typ sind frei kombinierbar

KAPITEL16 – Replikationsplanung und Implementierung

Design Considerations: what data (subset, group articles) who eceives, how often (schedule oder demand), network characteristics (node availability, how fast ~ need to filter,capacity ~ if low remote distributor, reliable ~ if not no distributed transactions)
Replikationslösung bestimmen: Topologie (Replikationsmodell wählen, D remote oder lokal, Distribution DB shared or not), Replikationstyp (S; T; M), wer initiiert Replikationsaktivität (PUSH/PULL), wie viele Subscriber (Schätzung wegen Distribution Server Load), Space requirements (Transaktionslogs, Größe Distribution DB und working folder)
Data Definition Issues: Replikationsdaten müssen mit speziellen Characteristics definiert werden (manche Charakteristiken können nicht oder werden verändert repliziert)
Replikationsdatentypen: **TIMESTAMP** (beinhaltet keine Zeitangabe, einzigartige Nummern, die Veränderungen in Reihen anzeigen, da für Subscriber ohne Relevanz können diese Reihen rausgefiltert werden, wird als binary data type repliziert, bei immediate update option notwendig, für merge müssen sie entfernt werden da sonst keine Datensammlung möglich), **UNIQUEIDENTIFIER** (Global **GUID**, wird Spalten über **NEWID()** Funktion zugewiesen, bei MERGE gebraucht), **USER-DEFINED** (kann nicht repliziert werden, wenn dieser Datentyp nicht schon in der Subscriber DB existiert)
Empfohlen: alle Replikationsserver sollten gleiches character Set und Sort Order benutzen **IDENTITY Eigenschaft**: Wert der Spalte mit IDENTITY Eigenschaft wird repliziert, nicht aber die Eigenschaft., d.h. Werte auf Subscriber matchen immer Publisherwerte. Um Daten zu partitionieren, muß Identityspalte auf Subs. erneut aufgesetzt werden, erst **CHECK**, dann Identity Spalte mit anderer verbinden, um modifizierte Daten eindeutig zu identifizieren
NOT FOR REPLICATION Option: Features ausstellen (IDENTITY, CHECK Blocker (constraints) und Trigger), geeignet i.V.m. Multiple P/Multiple S Modell
Netzwerksicherheitszugriff konfigurieren: bei verschiedenen Domänen **Trusties** einrichten, Replikation benutzt default NT Domain User account vom SQL Server Agent (alle Replikationsbeteiligten Server sollten dieses eine Account benutzten)

Server vorbereiten: D und P aufsetzen und Publikationen und Subskriptionen definieren
Distributor aufsetzen: **erst D dann P**, nur **sysadmin** kann, über **Configure Publishing and Distribution Wizard**, Replikationsserver müssen in **EM** registriert sein
DISTRIBUTION DB konfigurieren: store-and-forward DB mit allen wartenden Transaktionen, auto install bei Installation des D, Option ein remote D zu bestimmen oder neue DB auf konfiguriertem D einrichten, eine oder mehrere DBs bestimmen, Lokation der Daten und Logfiles angeben
Zugriff auf Distribution Working Folder feststellen: Zugriff von anderen Servern über default administrative share (**C$**) **\\computer_name\C$\Mssql7\Repldata** (eigenes share definieren geht auch), dieses share muß für Agent Service Account zugreifbar sein, auf **WIN9X** gibt es kein default share (deswegen anlegen, bei Benutzung der default **Publisher Konfiguration C$** share auf **C:root** anlegen und Berechtigungen hinzufügen)
Distributor konfigurieren: nachdem D enabled ist über ***P oder D Properties*** oder ***EM-Tools-Replication-Configure***: **DISTRIBUTION DATABASES** (name, file, retention properties ~ history und transaction records), **PWD** (für Publisher Verbindung), **AGENT**

PROFILES (für assoziierte Agents), **PUBLISHERS WÄHLEN** (für remote Verbindung, P werden D DB zugewiesen)
Änderung der D DB des P: P disablen → drop publications und subscriptions → P Server als neuen P enablen → nue DB zuweisen
Speicherplatz zusichern: bei TR werden auch Daten in der D DB gespeichert! Weitere Faktoren: Tabellenzahl, Articles, **Retention Time** (Transaktionsinfo wird beibehalten bei allen Subscribern erfolgreich beendet), bei TR zusätzlich: INSERT und UPDATE, geschätzte Transaktionsrate, durchschnittl. Transaktionsgröße
Distribution DB löschen: möglich, ohne D zu disablen, aber vorher alle Publisher disablen, die D DB benutzen
Uninstall Distributor: über Disable Publishing and Distribution Wizard, entfernt alle Replikationskomponenten vom Server, alle D DBs auf Server werden gelöscht, alle dazugehörenden P disabled und Publications gelöscht, Subscriptions to Publications werden gelöscht (aber keine Daten, die schon beim S sind)
→ Distributors brauchen viel RAM!

Publisher aufsetzen: wie D (gleiches Dialogfeld), **Property Dialog Box**: MR oder TR involvierte DBs spezifizieren, S enablen mit Sicherheit und default scheduling, Acces list of logins verwalten (nur für PULL und immediate-updating)

Subscriber aufsetzen: 2 Typen (registrierte und anonyme)
Registreirte S: Registrierung beim P gespeichert, Performance Info des S beim D
Anonyme S: keine detaillierten Infos bei P und D, Anwendung bei großer Anzahl von Subscribern oder Verbindung übers Internet, müssen beim P nicht enabled werden
Enabling S: beim P, zur Liste hinzufügen, Sicherheit und default agent schedules konfigurieren, überprüfen, ob gültiges Account beim D besteht
Disabling S: beim P, Subscriptions für alle Publikationen werden auto deleted, S Admin ist für Löschung der S DB selber verantwortlich
Agent Profiles: speichert Agent Eigenschaften, es gibt default profile oder custom, um administrativen Aufwand für Replikationsbatches, time-outs, polling intervals zu verringern
Handsymbol: unter Folder Icon von gepublishten DBs in ***EM-Server-Databases***, in dieser DB wurde ebenfalls eine neue Node mit Namen Publications hinzugefügt (hier sind alle Subscriptions aufgelistet)
Destination DB: hier neue Node Pull Subscriptions

Publishing:
3 **Methoden, Publications zu erstellen**: **Create Publishing Wizard**, ***EM-Tools-Replication-Create-Manage Publications***, **Replicate Data Taskpad**
Publikationen definieren: Typ (S, T, M), all Subscribers SQL Server?, Schedule, Filtering on Articles, what allow? (anonymous, updating, pull), share agent? (default jede Publikation hat ihren eigenen Agenten)
Referentielle Integrität: alle Tabellen mit relationship in publication einbeziehen
Publication ändern: neue Artikel etc. lassen sich hinzufügen, aber zuerst alle S entfernen
Articles spezifizieren: Wizard zeigt Tabellen, bei SR und TR auch Prozeduren, Artikeleigenschaften ändern über **Ellipsis** (...)
Articles bei MERGE REPL: Publikationen dürfen keine Timestamp-Spalte haben (clock icon ist in diesem Fall über der checkbox), vor Publikationserzeugung zuerst mit **ALTER DATABASE** statement entfernen
Filter einrichten: vertikaler Filter (Spalten wählen, geht nicht bei MERGE, Spalten müssen Primary Key beinhalten, SR braucht keinen Primary Key), horizontaler Filter (mit **WHERE** clause)
Filtering bei MERGE: unterstützt **dynamische Filterung** (S können unterschiedliche replizierte Daten einer Publikation empfangen)
Immediate Updating Subscribers Option: Option nicht bei MERGE, upzudatende Tabellen brauchen Timestampspalte

Publishing Considerations: Restriktionen (cannot span DBs; keine Replikation von MODEL, TEMPDB, MSDB, MASTER, DISTRIBUTION; Tabelle sollte Primary Key außer bei SR enthalten), limitierte Unterstützung für nonlogged Operations (kein Logging bei Benutzung der statements WRITETEXT, UPDATETEXT)