Datenbankimplementierung

Sofern Ausführungen auf SQL Server 2005 Bezug nehmen, gelten sie gleichermaßen auch für SQL Server 2008.

Christian Zahler

1 Datenbank-Grundlagen

Eine Datenbank ist eine Sammlung von Daten aus der Realität.

1.1 Arten von Datenbanken

1.1.1 Sequenzieller Zugriff

Älteres Datenzugriffsverfahren (Speicherung auf Magnetbändern!). Sequenziell = „hintereinander" (vgl. Videokassette).

- Datensätze haben Trennzeichen (etwa ANSI-13 = Zeilenumbruch); Datenfelder haben ebenfalls Trennzeichen (etwa Semikolon)
- Daten können nur sequentiell (nacheinander) gelesen werden. Daher ist dieses System extrem langsam beim Suchen und Sortieren (Man stellt am Vorabend eine Abfrage, die erst am nächsten Tag ausgewertet wird.)
- keine fixe Datensatzlänge, daher speicherplatzsparend

Beispiel

Das CSV-Dateiformat *(Comma Separated Value)* kann von Excel gelesen werden und wird oft als Schnittstelle zu Großdatenbanksystemen verwendet.

```
KNr;Nachname;Vorname;PLZ;Strasse
1;Camino;Alejandra;28001;Gran Vía, 1
2;Feuer;Alexander;04179;Heerstr. 22
3;Trujillo;Ana;05021;Avda. de la Constitución 2222
4;Domingues;Anabela;05634-030;Av. Inês de Castro, 414
5;Fonseca;André;04876-786;Av. Brasil, 442
6;Devon;Ann;WX3 6FW;35 King George
7;Roulet;Annette;31000;1 rue Alsace-Lorraine
8;Moreno;Antonio;05023;Mataderos  2312
9;Cruz;Aria;05442-030;Rua Orós, 92
10;Braunschweiger;Art;82520;P.O. Box 555
11;Batista;Bernardo;02389-673;Rua da Panificadora, 12
12;Schmitt;Carine;44000;54, rue Royale
13;González;Carlos;3508;Carrera 52 con Ave. Bolívar #65-98 Llano Largo
14;Hernández;Carlos;5022;Carrera 22 con Ave. Carlos Soublette #8-35
15;Dewey;Catherine;B-1180;Rue Joseph-Bens 532
```

Beispiel

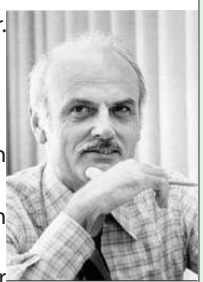
```
INI-Dateien
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server, Version
4.0"
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server, Version 4.0
[VGA-Modus]" /basevideo /sos
```

1.1.2 Index-sequentieller Zugriff

- Speicherung der Daten so wie beim sequentiellen Zugriff
- Zusätzlich wird eine „schlanke" Index-Datei angeegt, in der zum Beispiel ein „indiziertes Feld" (etwa der Nachname) und die Nummer des Bytes, an dem der Datensatz beginnt, gespeichert wird. Eine Suche nach Nachnamen ist somit wesentlich schneller möglich, da nur die Indexdatei durchsucht wird und nicht die gesamte Datenbank.

1.1.3 Relationales Konzept

Das relationale Datenbankmodell wurde 1970 von Dr. Edgar Frank CODD (1923 – 2003) entwickelt.



Dr. Edgar Frank "Ted" CODD

- Die Daten sind generell in Relationen gespeichert.
- Relationen sind Tabellen, wobei
  - die Reihenfolge der Spalten ("Felder") egal sein muss
  - die Reihenfolge der Zeilen (Datensätze) egal sein muss
  - es ein Feld geben muss, über dessen Wert jeder Datensatz eindeutig identifiziert werden kann ("Primärschlüssel")
- Relationen bestehen aus Feldern („Spaltenüberschriften"), deren konkrete Ausprägungen als „Attribute" (in Excel= Zelle) bezeichnet werden.
- Der Wertebereich eines Attributs kann eingeschränkt sein.

Marktübersicht für relationale Datenbank-Management-Systeme (RDBMS)

1. Dateibasierende Datenbanksysteme ("Klein-Datenbanken")

Bei diesen Datenbanksystemen befinden sich alle Datenbank-Objekte (Tabellen, Abfragen etc.) alle in einer einzigen Datei (zum Beispiel in Access: *.MDB-Datei).

- Microsoft Access, aktuelle Version Access 2003 (für "experience", intern Version 11)
- Microsoft FoxPro
- MySQL (Linux Open Source)

2. Client-/Server-Datenbanksysteme

Hier sind die Datenbankobjekte auf mehr als eine Datei verteilt. Typischerweise gibt es keine Berichts- und Formularobjekte. Der Server stellt benötigte Daten meist als "Datensatzgruppen" *(Recordsets)* den Clients zur Verfügung, die Darstellung wird meist am Client von Frontend-Software übernommen.

- Microsoft SQL Server
- Oracle
- postgreSQL (Linux Open Source)
- Sybase Adaptive Server
- Informix-Systeme

1.1.4 Hierarchische Datenbanken, XML-Datenbanken

Hier hat sich in den letzten Jahren XML einen fixen Platz in der Datenspeicherung erobert.

```
<?xml version="1.0" encoding="UTF-8"?>
<tKunden>
<datensatz>
  <KdNr>23</KdNr>
  <Vorname>Helmut</Vorname>
  <Nachname>Gruber</Nachname>
</datensatz>
<datensatz>
  <KdNr>47</KdNr>
  <Vorname>Maria</Vorname>
  <Nachname>Gschwandtner</Nachname>
</datensatz>
<datensatz>
  <KdNr>19</KdNr>
  <Vorname>Peter</Vorname>
  <Nachname>Maier</Nachname>
</datensatz>
</tKunden>
DTD
<!DOCTYPE tKunden [
  <!ELEMENT tKunden (datensatz+) >
  <!ELEMENT datensatz (KdNr,Vorname*,Nachname) >
  <!ELEMENT KdNr (#PCDATA) >
  <!ELEMENT Vorname (#PCDATA) >
  <!ELEMENT Nachname (#PCDATA) >
]>
XML-Schema
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
<xsd:element name="tKunden">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="datensatz" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="generated" type="xsd:dateTime"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="datensatz">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="KdNr" minOccurs="0" od:jetType="longinteger"
od:sqlSType="int" type="xsd:int">
<xsd:annotation>
<xsd:appinfo>
<od:fieldProperty name="ColumnWidth" type="3" value="-1"/>
<od:fieldProperty name="ColumnOrder" type="3" value="0"/>
<od:fieldProperty name="ColumnHidden" type="1" value="0"/>
<od:fieldProperty name="DecimalPlaces" type="2" value="255"/>
<od:fieldProperty name="Required" type="1" value="0"/>
```

```
<od:fieldProperty name="DisplayControl" type="3" value="109"/>
<od:fieldProperty name="TextAlign" type="2" value="0"/>
<od:fieldProperty name="AggregateType" type="4" value="-1"/>
</xsd:appinfo>
</xsd:annotation>
</xsd:element>
<xsd:element name="Vorname" minOccurs="0" od:jetType="text"
od:sqlSType="nvarchar">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="255"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="Nachname" minOccurs="0" od:jetType="text"
od:sqlSType="nvarchar">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="255"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

1.2 Datenbankplanung

1.2.1 Planung von Datenbanken; Entity-Relationship-Modell

- Welche Informationen gehören in die Datenbank und sollen gespeichert werden?
- Datenbankstruktur

Grafische Unterstützung beim DB-Design bietet das **Entity-Relationship**-Modell (Dr. Peter Chen, 1976).

Dr. Peter Chen

In diesem Modell sind folgende Begriffe wesentlich:

- **Entity**: Ein real existierendes Objekt, das in einer DB abgebildet werden soll. Wird durch ein Rechteck gekennzeichnet.
- **Relationship**: gibt an, wie zwei Entitäten miteinander verknüpft sind. Relationships werden durch eine Raute symbolisiert.
- **Attribut**: "Feld", wird durch ein Oval dargestellt
- **Primärschlüsselattribut**: Der Attributname wird zusätzlich unterstrichen.

Beispiel

Kardinalität von Beziehungen: Sie gibt an, wie viele Elemente der einen Entität mit wie vielen Elementen der anderen Entität in Beziehung stehen.

a) 1:1-Beziehung

Jedem Element der linken Entität kann nur genau ein Element der rechten Entität zugeordnet werden und umgekehrt.

Beispiel

b) 1:n-Beziehung

Jedem Element der linken Entität können beliebig viele Elemente der rechten Entität zugeordnet werden. Jedem Element der rechten Entität kann nur genau ein Element der linken Entität zugeordnet werden.

Beispiel

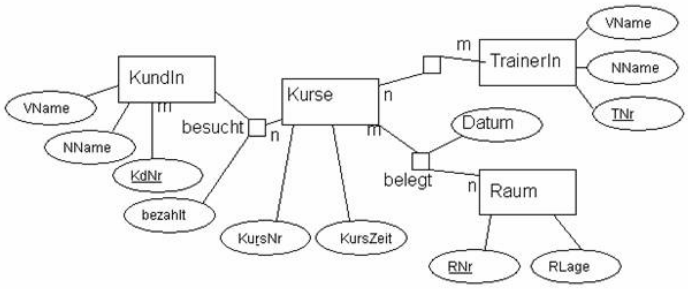
c) m:n Beziehung

Beliebig vielen Elementen der linken Entität können beliebig viele Elemente der rechten Entität zugeordnet werden. Dieser Verknüpfungstyp kommt in der Realität am häufigsten vor.

Beispiel

Hinweis: m:n Beziehungen können nicht direkt in ein relationales Modell übertragen werden.

Beispiel: ER-Diagramm für ein Schulungsinstitut



1.2.2 Umsetzung des ER-Diagramms in das relationale Modell

Hier sind nur einige Grundregeln zu beachten:

1. Aus jeder Entität wird eine Relation. Relationen dieser Art werden oft als "Stammdaten-Tabelle" bezeichnet.
2. Bei 1:1-Beziehungen ist zu überprüfen, ob die beiden Entitäten nicht in einer Tabelle zusammengefasst werden können.
3. 1:N-Beziehungen können direkt in ein relationales Modell umgesetzt werden; in die N-Tabelle muss ein Fremdschlüsselfeld eingefügt werden.
4. M:N-Beziehungen sind nicht direkt in ein relationales Modell umsetzbar; hier ist eine Zwischentabelle notwendig.

Die Umsetzung wird zu einem späteren Zeitpunkt an Hand des praktischen Beispiels näher erläutert.

1.3 Normalisierung von Datenbanken

Ziele bei der Realisierung von Datenbanken ist die Vermeidung von:

- **Redundanz**: Die Daten in einer Datenbank sind dann redundant, wenn Teile der Daten mehrfach vorkommen!
- **Inkonsistenz**: mehrere Schreibweisen für ein und dasselbe Objekt: zum Beispiel `St. Pölten, Sankt Pölten, St. Poelten, St Pölten, ...`

Zur Vermeidung von Redundanzen und Inkonsistenzen gibt es die so genannten Normalformen. Wenn die Tabellen einer DB den Normalformen genügen, ist ein wichtiger Beitrag zur Redundanzvermeidung geleistet (noch keine Garantie, dass überhaupt keine Redundanz!)

- **1. Normalform**: Keine Listen als Wertebereiche
- **2. Normalform**: Attribute dürfen nicht von einem Teil eines Schlüssels abhängen
- **3. Normalform**: Attribute dürfen nicht voneinander ableitbar sein.

1.3.1 1. Normalform

- Jedes Feld besitzt einen eindeutigen Namen, kein Feld kommt mehrfach vor.
- Jedes Datenelement ist atomar und nicht weiter zerlegbar.
- Jede Tabelle besitzt einen Primärschlüssel.
- Beziehungen zwischen Entitäten werden ausschließlich über Schlüsselfelder hergestellt (keine absoluten Adressen).

Probleme bei Datenbanken in 1. Normalform

- Identische Attributwerte werden mehrfach gespeichert (Redundanz)
- Einfügeanomalien (Es kann kein Student angelegt werden, der sich noch nicht für ein Seminar entschieden hat)
- Löschanomalien (Student muss gelöscht werden, falls er alle gebuchten Seminare absagt)
- Änderungsanomalien (Eine nachträgliche Änderung der Attribute (Namensänderung bei Heirat) führt zu Änderung an mehreren Datensätzen)

Als Attributwerte sind nur atomare Werte (`integer`, `string`) erlaubt, keine Listen oder Mengen.

tblBuch		
Buchnr	Buchtitel	Autor
184	Sozialstaat Österreich	Ernst,Federspiel,Langbein

Lösung

tblBuch		
Buchnr	Buchtitel	Autor
184	Sozialstaat Österreich	Ernst
184	Sozialstaat Österreich	Federspiel
184	Sozialstaat Österreich	Langbein

tblKunden	
Name	Adresse
Harrer, Heinrich	Bahnhofplatz 3, 3100, St. Pölten

Lösung: Zerlegung in mehrere Felder

Nachname	Vorname	Straße	PLZ	Ort
Harrer	Heinrich	Bahnhofplatz 3	3100	St.Pölten

1.3.2 2. Normalform

- Die Entität liegt in 1. Normalform vor.
- Jedes Feld, das nicht Bestandteil des Primärschlüssels ist, ist voll funktional abhängig vom Primärschlüssel (alle Teile des Primärschlüssels werden benötigt, um die restlichen Felder zu bestimmen), d.h. Beseitigung der nicht voll funktionalen Abhängigkeiten.

Andere Formulierung

Eine Tabelle befindet sich in der 2. Normalform, wenn

a) sie sich in der 1. Normalform befindet und wenn

b) alle Nichtschlüsselattribute von allen Attributen des Primärschlüssels abhängen.

Beispiel

tblEntlehnung					
Kundennr	Nachname	Vorname	.....	Buchnr	Entlehndatum
23	Müller	Aloisia		770182	02.05.2001
23	Müller	Aloisia		912341	02.05.2001
109	Giger	Brunhilde		891021	30.04.2001
176	Huber	Herbert		NULL	NULL

entspricht nicht der 2. Normalform:

`Buchnr` hängt nicht vom Primärschlüssel `Kundennr` ab

`Entlehndatum` hängt nicht vom Primärschlüssel `Kundennr` ab

Anomalien

1. Löschanomalie

Bei Rückgabe aller Bücher werden auch die Informationen über den/die Entleiher/in gelöscht.

2. Einfügeanomalie

Will man Informationen über einen Kunden einfügen, der noch kein Buch ausgeliehen hat, dann müssen alle Felder, die sich auf das Ausleihen von Büchern beziehen, mit `NULL`-Einträgen bzw. (noch schlimmer) mit Dummy-Einträgen gefüllt werden (z.B. `Buchnr = 999999` bedeutet *"noch kein Buch ausgeborgt"*). Setzt man Primärschlüssel auf `Buchnr`, so können Kunden, die noch kein Buch entlehnt haben, gar nicht angelegt werden. Ist allerdings `Kundennr` Primärschlüssel, so kann jeder Kunde nur ein Buch ausborgen.

3. Änderungsanomalie

Bei Änderung von Personendaten (neuer Name, neue Adresse, neue Telefonnummer) müssen diese Änderungen in mehreren Datensätzen durchgeführt werden. Wird ein betroffener Datensatz nicht geändert, so enthält die Datenbank widersprüchliche Informationen.

Lösung

Tabelle muss in mehrere Tabellen aufgespalten werden.

1.3.3 3. Normalform

- Jede Entität liegt in 2. Normalform vor.
- Jedes Feld, welches nicht Bestandteil des Kandidatenschlüssels ist, hängt nicht transitiv von einem Kandidatenschlüssel ab.

2 Transact SQL

Die Programmiersprache SQL *(Structured Query Language, Standard Query Language)* wurde von IBM entwickelt ? "Projekt R". Um 1975 kam man auf die Idee, für Datenbankabfragen eine eigene Programmiersprache zu entwickeln.

ca. 1977 – 79: Entwicklung von SQL (Programmiersprache der vierten Generation)

Generation 1: reine Maschinensprachen (binäre Programmierung)

Generation 2: Assembler

Generation 3: höhere Programmiersprachen

Generation 4: Datenbankabfragesprachen

SQL-Normen (ANSI = *American National Standards Institut*, ISO = *International Standardization Organization*)

- SQL-89 (ältere Version)
- SQL-92: ab Access 97 bzw. SQL Server 7.0
- SQL-99 (aktuelle Version): Kaum praktische Implementierungen vorhanden.

Viele Hersteller verwenden zusätzlich zu den ANSI-kompatiblen Basis-SQL-Befehlen produktspezifische Erweiterungen:

- TSQL *(Transact SQL)*: SQL-Erweiterung für Microsoft SQL Server
- SQL*Plus: Erweiterung für Oracle-Datenbanken

2.1 Erstellen einer SQL Server-Datenbank

Wiederherstellungsmodell *(Recovery Model)*

SQL 2000/2005	Bedeutung
Full	Log enthält alle Transaktionen seit dem letzten Backup; Log-File wird kontinuierlich wachsen
Simple	nur aktive Transaktionen sind im Log; Logfile sehr klein; kein Point-in-Time-Recovery, keine vollständige Datenwiederherstellung möglich
Bulk_Logged	erlaubt unprotokollierten Massenimport; andere Transaktionen werden jedoch protokolliert; kein Point-in-Time-Recovery

Verkleinern der Datenfiles

```
DBCC SHRINKDATABASE
DBCC SHRINKFILE
```

Optionen

`NOTRUNCATE` – Datenfile wird bis zur "Hochwassermarke" verkleinert

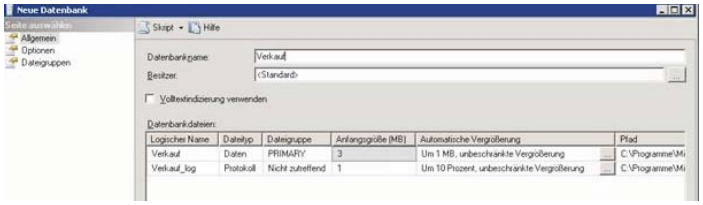
`EMPTYFILE` – Alle Daten dieses Files werden in andere Datenfiles verschoben

`TRUNCATEONLY` – Datenfile wird verkleinert, ohne die Daten intern zu verschieben

Dateigruppen

Werden verwendet, um die Flexibilität und Performance zu erhöhen. Tabellen werden am besten zunächst Dateigruppen zugeordnet, erst die Dateigruppe wird mehreren Datendateien zugeordnet.

Dateigruppe in den Datenbankeigenschaften anlegen; im Karteireiter "Data Files" können die einzelnen Datendateien einer Dateigruppe zugeordnet werden.



TSQL-Code

```
/* Anlage einer neuen Datenbank
   Skript Version 1.0
   11.05.2007  */
create database Verkauf
on primary    – Dateigruppe primary
  (name = 'verkauf1',filename='E:\verkauf1.mdf',
   size=10 MB,maxsize=unlimited,filegrowth=10 %),
filegroup daten2006  – weitere Dateigruppe, optional!
  (name = 'verkauf2',filename='E:\verkauf2.ndf',
   size=5 MB,maxsize=100 MB,filegrowth=10 MB)
log on        – Transaktionsprotokoll
  (name = 'verkauf_log',filename='F:\verkauf_log.ldf',
   size=2 MB,maxsize=unlimited,filegrowth=1 MB);
```

Datenbankeigenschaften ändern

```
ALTER DATABASE SampleDBTsql
  MODIFY FILE
  (NAME = 'SampleDBTsql_Log',
   MAXSIZE=20MB)
GO
```

Datenbanken löschen

```
USE master
DROP DATABASE SampleDBTsql, SampleDBWizard
GO
EXEC sp_helpdb
GO
```

2.2 Dateimäßiger Aufbau einer SQL Server 2005-Datenbank:

- **Hauptdatendatei** (Endung `*.MDF` = *main data file*): enthält die konkreten Datenbankobjekte, zum Beispiel Tabellen, Sichten, gespeicherte Prozeduren etc.; enthält Systemtabellen
- **weitere Datendateien** (`*.NDF` = *non-main data file*)
- **Transaktionsprotokoll** (`*.LDF` = *Transaction Log*): Alle Änderungen der Daten seit dem letzten Backup werden im Transaktionsprotokoll gespeichert. Dadurch werden Wiederherstellungen bis zum aktuellen Datenbestand möglich.

Die Datendateien und Transaktionsprotokolle sollten auf unterschiedlichen Laufwerken gespeichert werden.

Das Transaktionsprotokoll wird in einem internen Format gespeichert.

Ein "Checkpoint"-Prozess löst (etwa ein Mal jede Sekunde) die konkrete Aktualisierung der Datenbank auf der physischen Festplatte aus.

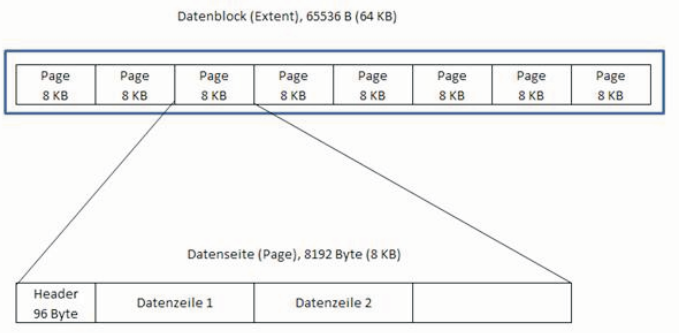
2.3 Interner Aufbau einer Datendatei:

Eine SQL Server-Datendatei (`*.mdf, *.ndf`) besteht grundsätzlich aus 8 KB großen Seiten (engl. *Pages*), in denen Daten gespeichert sind. Jeweils 8 aufeinanderfolgende Seiten bilden einen Block (engl. *Extent*).

Man unterscheidet:

- **Einheitlicher Datenblock**: alle 8 Seiten gehören zum selben Datenbankobjekt (etwa zur selben Tabelle)
- **Gemischter Datenblock**: die 8 Seiten gehören zu unterschiedlichen Datenbankobjekten.

Jede Seite beginnt mit einem 96 Byte großen Header und enthält dann einen oder mehrere Datenzeilen. Am Ende jeder Seite befindet sich die Zeilenoffsettabelle (32 Byte), die den „Abstand" jeder Datenzeile vom Beginn der Seite enthält, und zwar in umgekehrter Reihenfolge der Datenzeilen. Für die eigentliche Datenspeicherung stehen pro Seite max. 8060 Byte zur Verfügung.



Man unterscheidet nach dem Inhalt der Pages:

Datenseiten: Diese enthalten unterschiedliche Arten von Daten:

- **Datenseiten**: enthalten Daten (ausgenommen solche vom Datentyp `text`, `ntext`, `image`, `varchar(max)`, `nvarchar(max)`, `varbinary(max)`)
- **LOB-Seiten** (LOB = *Large Objects*): enthalten Daten vom Datentyp `text`, `ntext`, `image`, `varchar(max)`, `nvarchar(max)`, `varbinary(max)`
- **Indexseiten**

Verwaltungsseiten: Diese enthalten Informationen über die Struktur und den Aufbau der Datendatei.

- **GAM/SGAM** *(Global Allocation Map, Secondary GAM)*

GAM/SGAM-Seiten enthalten Informationen über den Zustand von 64000 Extents mit folgender Belegung:

	GAM-bit	SGAM-bit
Freier Datenblock (nicht in Verwendung)	1	0
Einheitlicher Datenblock oder voller gemischer Block	0	0
Gemischter Datenblock mit freien Seiten	0	1

- **PFS** *(Page Free Space)*

PFS-Seiten enthalten Informationen darüber, wie viel Platz auf den nächsten 8000 Blöcken noch frei ist.

- **IAM** *(Index Allocation Map)*

IAM-Seiten verwalten Indizes.

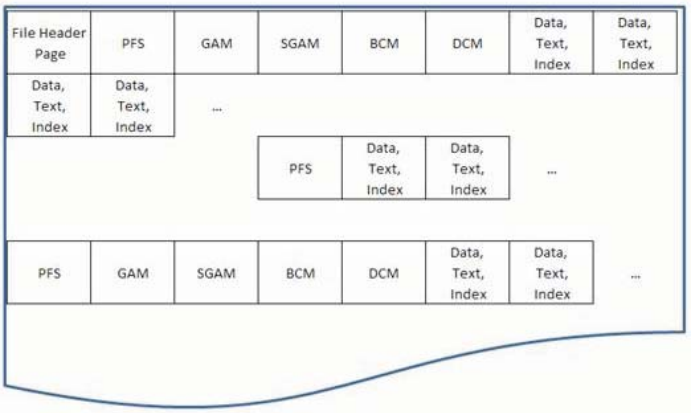
- **BCM** *(Bulk Changed Map)*

BCM-Seiten enthalten Informationen darüber, welche der nächsten 64000 Blöcke mit bcp seit dem letzten BACKUP LOG-Vorgang verändert wurden (wenn ein Bit auf 1 gesetzt ist, dann wurde der entsprechende Block geändert).

- **DCM** *(Differential Changed Map)*

BCM-Seiten enthalten Informationen darüber, welche der nächsten 64000 Blöcke seit dem letzten BACKUP LOG-Vorgang verändert wurden (wenn ein Bit auf 1 gesetzt ist, dann wurde der entsprechende Block geändert).

Jede Datendatei ist folgendermaßen aufgebaut:



Aufbau eines Datensatzes:

- Datensatz-Header
  - 4 Byte
  - 2 Byte Datensatz-Metadaten (record type)
  - 2 Byte, die auf das NULL-Bitmap zeigen
- Spalten mit fixer Länge (Datentypen zum Beispiel bigint, char(10), datetime)
- NULL-Bitmap
  - 2 Byte, die die Anzahl der Spalten im Datensatz speichern
  - variable Anzahl von Byte; je ein Bit pro Spalte wird verwendet, um anzugeben, ob die Spalte NULL enthält oder nicht (SQL Server 2000 verwendete nur Bits für Spalten, die NULL-Werte enthalten durften; in SQL Server 2005 wird ein bit für jede Spalte verwendet, egal, ob sie NULL-Werte annehmen darf oder nicht)
  - this allows an optimization when reading columns that are NULL
- Offset-Struktur für Spalten mit variabler Länge
  - 2 Byte, die die Anzahl der Spalten mit variabler Länge angeben
  - 2 Byte pro Spalte mit variabler Länge, die die Anzahl der Byte vom Beginn des Datensatzes bis zum Beginn der Spalte angibt (Offset)
- Versionsangabe (nur SQL Server 2005)
  - 14 Byte-Struktur, die einen Zeitstempel sowie einen Zeiger auf den Versionsspeicher in tempdb enthält.

2.4 Tabellen anlegen

Da die Daten in Tabellen gespeichert werden, werden als nächster Schritt neue Tabellen erstellt.

2.4.1 Regeln für Feldnamen, Tabellennamen und anderen Datenbank-Objekten

Feldnamen und andere Objektnamen dürfen **maximal 128 Zeichen** enthalten.

Verboten sind: Rufzeichen, eckige Klammern, Punkte und Akzentzeichen

Dringend abzuraten ist von der Verwendung von Leerzeichen, Umlauten und Sonderzeichen.

Erfüllen Feldnamen diese Regel, so werden sie **„reguläre Bezeichner"** genannt.

Dringend abzuraten ist von der Verwendung von Bezeichnungen, die bereits Access-intern verwendet werden, zum Beispiel „Name".

Wenn Sie einen Feldnamen wählen, der mit „-nummer" endet, so schlägt Access automatisch eine Indizierung *„Ja (Duplikate möglich)"* vor.

Tipp für das Speichern von Tabellen: Beginnen Sie den Namen der Tabelle mit einem kleingeschriebenen `t` oder `tbl` (also beispielsweise `tKunden`, `tblKunden`, `tbl_Kunden`); bei Abfragen verwenden Sie `q` (für *"query"*). Damit können Sie beim Erstellen von Formularen und Berichten Tabellen sofort von Abfragen unterscheiden.

2.4.2 Felddatentypen

a) Ganzzahl: int, smallint, bigint, ...

Mit ganzzahligen Werten kann exakt (ohne Ungenauigkeiten) gerechnet werden.

b) Dezimalzahl

`float()` *approximate numeric*

- Damit kann nicht exakt gerechnet werden. Es treten bei jedem Rechenvorgang Ungenauigkeiten und Rundungsfehler auf (z.B. 2.0 + 3.0 = 4.999999542)

`decimal()` *exact numeric*

- Exaktes Rechnen möglich, da skalierte Ganzzahl gespeichert wird.

c) Alphanumerisch

`char()` fixe Länge (max. 8000 Zeichen)

`varchar()` variable Länge (max. 8000 Zeichen)

`text()` lange Textfelder (max. 2 Mio. Zeichen)

`nchar()`, `nvarchar()`, `ntext()` ... *national character support*; hier wird UNICODE (2 Byte/Zeichen) verwendet, dies reduziert natürlich die Anzahl der verwendbaren Zeichen auf die Hälfte.

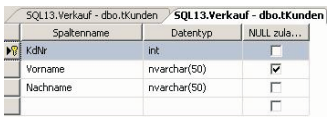
d) Datum/Zeit

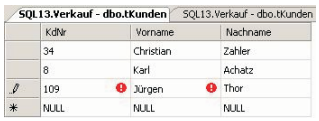
`datetime` ab der gregorianischen Kalenderreform bis 9999

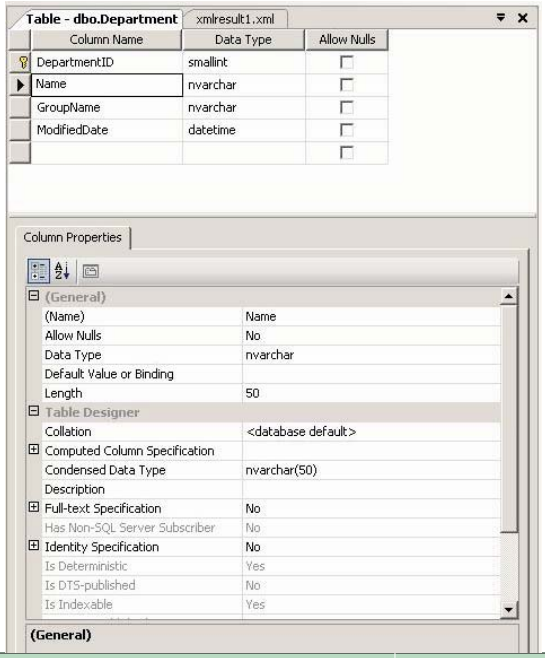
`smalldatetime` ab 1.1.1900 bis etwa 2090

`money` *Currency* - skalierte Ganzzahl; intern wird die Zahl mit 10000 multipliziert, für die Darstellung wieder dividiert und auf zwei Stellen gerundet.

2.4.3 Tabellen anlegen im Management Studio







2.4.4 Tabelle anlegen mit TSQL-Kommandos

```
use Auftrag;
create table dbo.tArtikel
( ArtNr int identity(1,1) primary key,
  ArtBez nvarchar(50) NOT NULL,
  Einzelpreis money NOT NULL
);

use Auftrag;
create table dbo.tAuftrag
(
  AuftrNr int identity(1,1) primary key,
  KdNr int not null,
  Datum datetime not null
)
```

2.5 Primärschlüssel und Indizes

Als Primärschlüssel wird ein Feld oder eine Kombination von Feldern verwendet, über die jeder Datensatz eindeutig identifizierbar ist. Die Werte des Primärschlüssels müssen also eindeutig sein, sodass aus der Kenntnis des Primärschlüsselwertes auf genau einen Datensatz rückgeschlossen werden kann.

Beispiele für Primärschlüsselfelder

- Kundennummer
- Artikelnummer
- Buchungsnummer

Es gibt auch mehrteilige Primärschlüssel. Dieser wird beispielsweise aus der Kombination von Geburtsdatum und Sozialversicherungsnummer gebildet wird. Dazu markiert man beide Zeilen mit gedrückter STRG -Taste. In diesem Fall Achtung: Die beiden Schlüsselsymbole sind irreführend; auch in dieser Tabelle gibt es nur **einen** Primärschlüssel!

Anmerkung: Jedes indizierte Feld kann als „Sekundärschlüssel" bezeichnet werden. Der Begriff wird aber im Zusammenhang mit Datenbanken praktisch nicht verwendet, da – außer den bereits erwähnten Zeitvorteilen beim Suchen und Sortieren – ein Sekundärschlüssel keine weiteren Vorteile bringt.

2.6 Auswahlabfragen

```
use AdventureWorks
select loginid,gender from HumanResources.Employee
```

2.6.1 Formulieren von Kriterien

Im der `WHERE`-Klausel werden nun Kriterien formuliert, nach denen die Daten gefiltert werden.

Textfelder

Suchmuster für Textfelder werden in Access immer mit einem doppelten Anführungszeichen gekennzeichnet.

`='Müller'`	exakte Übereinstimmung wird gefordert (es werden alle Datensätze im Abfrageergebnis ausgegeben, deren Eintrag im Nachnamen exakt dem Wort `„Müller"` entspricht); das =-Zeichen kann weggelassen werden
`LIKE 'S%'`	Wie = Ähnlichkeitsoperator; es sind auch Jokerzeichen im Suchmuster zugelassen. * ... 0 bis beliebig viele Zeichen (laut Norm: %)
`Wie 'M_er'`	? ... exakt ein unbekanntes Zeichen (laut Norm: _)
`<'S%'`	A bis R
`>'S%'`	S bis Z; eigentlich >= (>= gibt es bei Texten nicht)
`Between 'B%' And 'S%'`	B bis R

```
use AdventureWorks
select ProductID, Name
from Production.Product
where Name Between 'L%' And 'T%';
/* Ergebnis: L - S */
```

Datumsfelder

Einträge in Datumsfeldern werden in SQL Server so wie Texte gekennzeichnet.

`Between '01/01/1970' And '31/03/1970'`

`<Date()`

Zahlenfelder

Werte werden immer ohne spezielle Kennzeichnung (Anführungszeichen bzw. Nummernzeichen) eingetragen.

`Between 100 And 500` `<100 And >500`
`>34`
`<150`
`>=56,3`

Beim `Between`-Operator werden bei Zahlenfeldern beide Grenzen mit einbezogen.

Leere bzw. nicht-leere Felder

Nicht benützte Felder werden intern durch die symbolische Konstante `<NULL>` gekennzeichnet. `<NULL>`-Einträge können mit keinem anderen Wert verglichen werden, nicht einmal mit anderen `<NULL>`-Werten. Daher darf der `LIKE`-Operator in diesem Fall nicht verwendet werden, es gibt eine eigene Syntax:

`Is NULL`

`Is NOT NULL`

Kombination mehrerer Kriterien

Diese erfolgt mit BOOLEschen Operatoren (`AND`, `OR`, `NOT`)

a) Verknüpfung mit `AND`:

Dabei stehen Kriterien in derselben Zeile nebeneinander. Ein Datensatz erscheint nur dann im Abfrageergebnis, wenn beide Kriterien wahr sind. Übliche Darstellung: „Wahrheitswerte-Tabelle"

A	B	A And B
wahr	wahr	wahr
wahr	falsch	falsch
falsch	wahr	falsch
falsch	falsch	falsch

b) Verknüpfung mit `OR` (nicht ausschließendes ODER):

Der Datensatz erscheint dann im Abfrageergebnis, wenn mindestens ein Kriterium oder auch beide wahr sind. Achtung: Es entspricht nicht dem üblichen Sprachgebrauch. Dabei steht das erste Kriterium steht in der Kriterienzeile und das zweite Kriterium steht in der Oder-Zeile.

A	B	A Or B
wahr	wahr	wahr
wahr	falsch	wahr
falsch	wahr	wahr
falsch	falsch	falsch

2.6.2 Berechnete Felder in Abfragen

```
select anzahl, einzelpreis, anzahl * einzelpreis as gesamtpreis
From dbo.Artikel
```

2.6.3 Aggregatfunktionen und Gruppierung:

Beispiel

```
use adventureWorks
select ProductLine, sum(StandardCost) as Gruppensumme
from Production.Product
group by ProductLine;
```

Ergebnis

```
ProductLine Gruppensumme
----------- ---------------------
NULL        2115,0796
M           41125,5062
R           59497,1694
S           858,0118
T           26740,1255
(5 Zeile(n) betroffen)
```

Beispiel

```
use adventureWorks
select ProductLine, sum(StandardCost) as Gruppensumme
from Production.Product
where ProductLine = 'M'
group by ProductLine;
```

Ergebnis

```
ProductLine Gruppensumme
----------- ---------------------
M           41125,5062
(1 Zeile(n) betroffen)
```

Beispiel

Soll nach aggregierten Werten gefiltert werden, darf `WHERE` nicht verwendet werden. Statt dessen muss eine `HAVING`-Klausel nach der `GROUP BY`-Klausel angefügt werden.

```
use adventureWorks
select ProductLine, sum(StandardCost) as Gruppensumme
from Production.Product
group by ProductLine
having sum(StandardCost) > 20000;
```

Ergebnis

```
ProductLine Gruppensumme
----------- ---------------------
M           41125,5062
R           59497,1694
T           26740,1255
(3 Zeile(n) betroffen)
```

Beispiel

```
use adventureWorks
select ProductLine, sum(StandardCost) as Gruppensumme
from Production.Product
group by ProductLine
with rollup;
```

Ergebnis

```
ProductLine Gruppensumme
----------- ---------------------
NULL        2115,0796
M           41125,5062
R           59497,1694
S           858,0118
T           26740,1255
NULL        130335,8925
(6 Zeile(n) betroffen)
```

Hier kann man nicht eindeutig erkennen, welche Zeile die Gesamtsumme über alle Gruppensummen darstellt.

```
use adventureWorks
select ProductLine, sum(StandardCost) as Gruppensumme,
  grouping(ProductLine) as IstSumme
from Production.Product
group by ProductLine
with rollup;
```

```
ProductLine Gruppensumme          IstSumme
----------- --------------------- --------
NULL        2115,0796             0
M           41125,5062            0
R           59497,1694            0
S           858,0118              0
T           26740,1255            0
NULL        130335,8925           1
(6 Zeile(n) betroffen)
```

Im folgenden Beispiel werden die Gruppensummen für ProductLine und `ProductSubCategoryID` berechnet. Der `CUBE`-Operator berechnet zusätzlich die Summen pro `ProductLine` und pro `ProductSubCategoryID`.

```
select
 productline,
 ProductSubCategoryID,
 sum(StandardCost) as Gruppensumme,
 grouping(productline) as IstAggrProductLine,
 grouping(ProductSubCategoryID) as IstAggrSubCat
 from production.product
 group by productline ,ProductSubCategoryID
 with cube;
```

```
productline
           ProductSubCategoryID
                       Gruppensumme
                                   IstAggrProductLine
                                                IstAggrSubCat
-------------------------------------------------------------
NULL       NULL         1060,89     0          0
NULL       5            122,8638    0          0
NULL       6            94,572      0          0
NULL       7            8,9866      0          0
NULL       8            371,6148    0          0
NULL       9            94,5498     0          0
NULL       10           245,6209    0          0
NULL       11           115,9817    0          0
NULL       NULL         2115,0796   0          1
M          1            29880,6408  0          0
M          4            100,6682    0          0
M          12           10218,2052  0          0
M          13           81,5052     0          0
M          15           52,7917     0          0
M          17           542,1108    0          0
M          20           47,0127     0          0
M          22           78,5289     0          0
M          23           6,7926      0          0
M          27           59,466      0          0
M          28           3,7363      0          0
M          30           8,2205      0          0
M          36           10,3084     0          0
M          37           35,5189     0          0
M          NULL         41125,5062  0          1
R          2            42721,6278  0          0
R          4            100,6682    0          0
R          13           81,5052     0          0
R          14           15852,432   0          0
R          15           35,4135     0          0
R          17           625,6048    0          0
```

```
R          23          6,7246          0          0
R          28          3,3623          0          0
R          33          38,7627         0          0
R          37          31,0683         0          0
R          NULL        59497,1694      0          1
S          18          111,3627        0          0
S          19          6,9223          0          0
S          20          27,4779         0          0
S          21          320,2584        0          0
S          22          98,9836         0          0
S          24          92,8002         0          0
S          25          71,247          0          0
S          26          44,88           0          0
S          28          1,8663          0          0
S          29          2,9733          0          0
S          31          39,2589         0          0
S          32          20,5663         0          0
S          34          10,3125         0          0
S          36          8,2459          0          0
S          37          0,8565          0          0
S          NULL        858,0118        0          1
T          3           19490,5544      0          0
T          4           61,1211         0          0
T          13          35,9596         0          0
T          15          70,1699         0          0
T          16          6812,4686       0          0
T          17          205,5808        0          0
T          35          51,5625         0          0
T          37          12,7086         0          0
T          NULL        26740,1255      0          1
NULL       NULL        130335,8925     1          1
NULL       NULL        1060,89         1          0
NULL       1           29880,6408      1          0
NULL       2           42721,6278      1          0
NULL       3           19490,5544      1          0
NULL       4           262,4575        1          0
NULL       5           122,8638        1          0
NULL       6           94,572          1          0
NULL       7           8,9866          1          0
NULL       8           371,6148        1          0
NULL       9           94,5498         1          0
....
(99 Zeile(n) betroffen)
```

Die Klausel `COMPUTE` liefert Ergebniszeilen in einem nicht-relationalen Format. Nicht empfehlenswert!

```
select name, productline,StandardCost,ProductSubCategoryID
 from production.product
  order by productline, ProductSubCategoryID
 compute sum(StandardCost) by productline
compute sum(StandardCost);
```

Ergebnis

```
….
Touring Rear Wheel          T          108,7844          17
Touring Front Wheel         T          96,7964           17
Touring-Panniers, Large     T          51,5625           35
Touring Tire Tube           T          1,8663            37
Touring Tire                T          10,8423           37
sum
---------------------
26740,1255
sum
---------------------
130335,8925
(510 Zeile(n) betroffen)
```

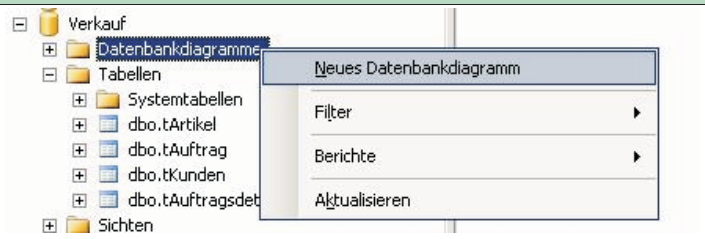
2.7 Beziehungen in Diagrammen erstellen

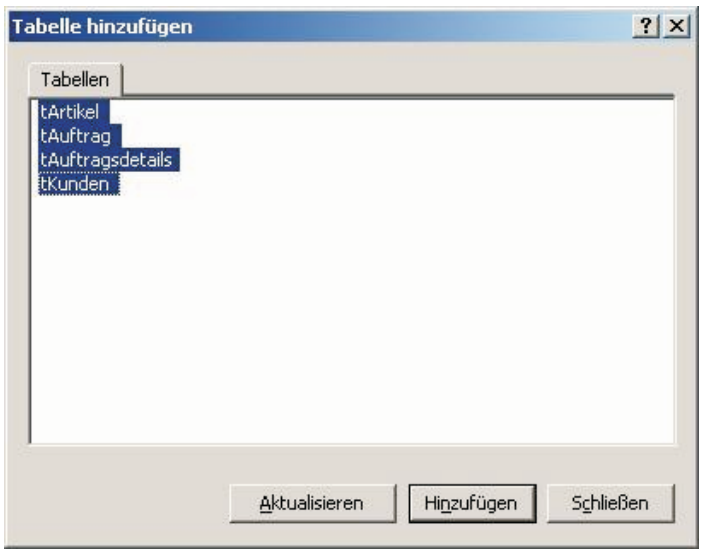
Die Erstellung von Fremdschlüsseleinschränkungen (Beziehungen) aktiviert einen Mechanismus im SQL Server, der die Integrität der eingegebenen Daten prüft (referentielle Integrität).

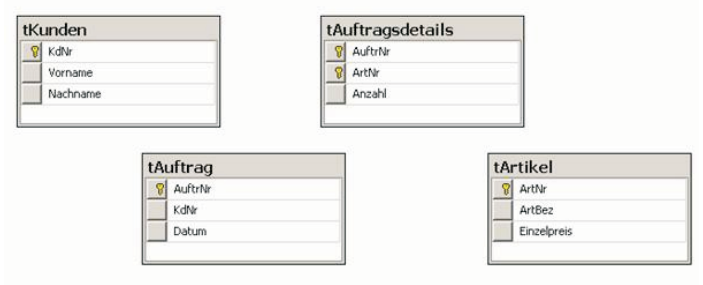
Referenzielle Integrität: Beispielsweise dürfen in einer Verkaufstabelle nur Kunden enthalten sind, die auch in der Kundentabelle angelegt sind

- Prüfung, ob Fremdschlüsselfelder vorhandenen Primärschlüsselfeldern entsprechen
- aus Mastertabelle können Datensätze (Tupel) erst dann gelöscht werden, wenn die verknüpften Datensätze in der Detailtabelle gelöscht werden

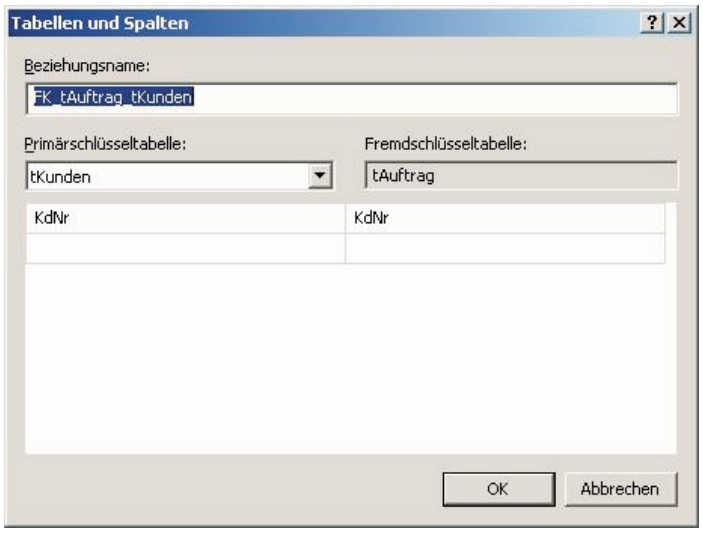
Im SQL Server 2005 Management Studio müssen dafür Datenbankdiagramme erstellt werden.

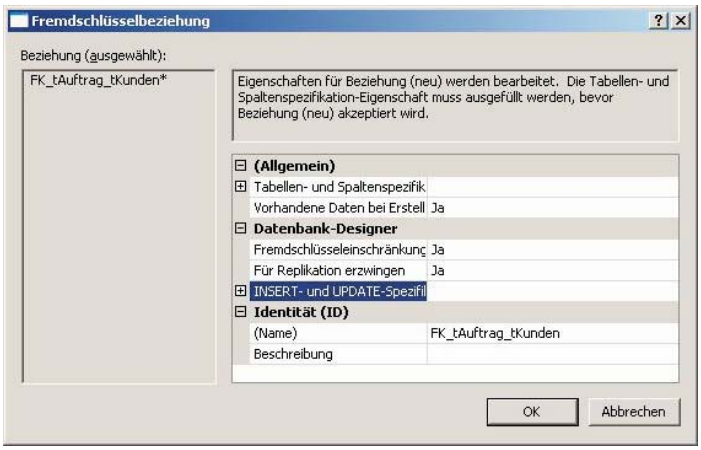


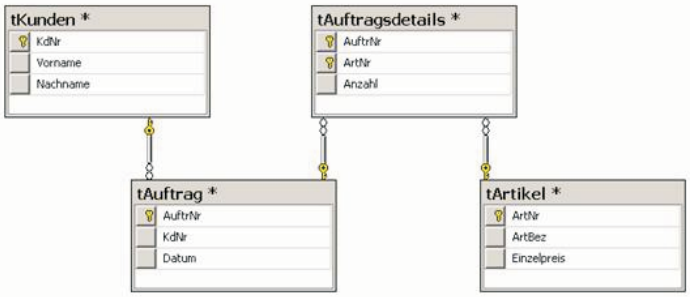




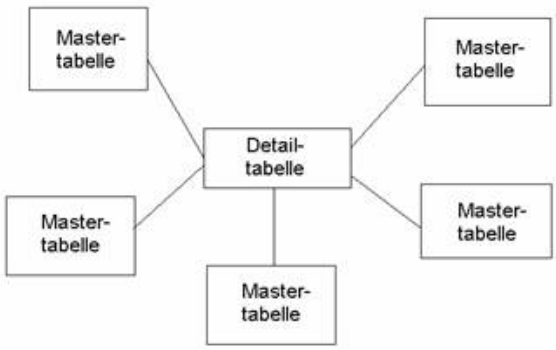
Die Beziehung wird erstellt, in dem die verknüpften Felder mit Drag and Drop verbunden werden.







In einem Diagramm ist es meist günstig, wenn man die Mastertabellen (Stammdatentabellen mit Primärschlüsseln) sternförmig um die Detailtabellen (Fremdschlüsseltabellen) anordnet:



2.8 Auswahlabfragen basierend auf mehreren Tabellen

```
select
  tAuftragsdetails.AuftrNr,
  tAuftrag.Datum,
  tAuftrag.KdNr,
  tKunden.Vorname,
  tKunden.Nachname,
  tAuftragsdetails.ArtNr,
  tArtikel.ArtBez,
  tAuftragsdetails.Anzahl,
  tArtikel.Einzelpreis,
  Anzahl * Einzelpreis AS Zeilenpreis
from
  tKunden inner join tAuftrag
    on tKunden.KdNr = tAuftrag.KdNr
  inner join tAuftragsdetails
    on tAuftrag.AuftrNr = tAuftragsdetails.AuftrNr
  inner join tArtikel
    on tAuftragsdetails.ArtNr = tArtikel.ArtNr
where
  tAuftragsdetails.AuftrNr = 1;

Aliasnamen:

select
  tAuftragsdetails.AuftrNr,
  tAuftrag.Datum,
  tAuftrag.KdNr,
  tKunden.Vorname,
  tKunden.Nachname,
  tAuftragsdetails.ArtNr,
  tArtikel.ArtBez,
  tAuftragsdetails.Anzahl,
  tArtikel.Einzelpreis,
  Anzahl * Einzelpreis AS Zeilenpreis
from
  tKunden as k inner join tAuftrag as a
    on k.KdNr = a.KdNr
  inner join tAuftragsdetails  as d
    on a.AuftrNr = d.AuftNr
  inner join tArtikel as art
    on d.ArtNr = art.ArtNr
where
  d.AuftrNr = 1;
```

CROSS JOINS

```
use Verkauf
select tKunden.Nachname, tArtikel.ArtBez
from tKunden, tArtikel
```

liefert CARTESISCHES Produkt von `tKunden` x `tArtikel`

```
Nachname                       ArtBez
---------------------------------------------
Fröschl                        Socken schwarz
Achatz                         Socken schwarz
Zahler                         Socken schwarz
Wehba                          Socken schwarz
Thor                           Socken schwarz
Keil                           Socken schwarz
Moser                          Socken schwarz
Fröschl                        T-Shirt rot
Achatz                         T-Shirt rot
Zahler                         T-Shirt rot
Wehba                          T-Shirt rot
Thor                           T-Shirt rot
Keil                           T-Shirt rot
Moser                          T-Shirt rot
Fröschl                        Socken blau
Achatz                         Socken blau
Zahler                         Socken blau
Wehba                          Socken blau
Thor                           Socken blau
Keil                           Socken blau
Moser                          Socken blau
(21 Zeile(n) betroffen)
```

Verknüpfungen mit derselben Tabelle:

```
use Auftrag
select
  a.Vorname, a.Nachname, 'Chef:', b.Vorname, b.Nachname
from
  tMitarbeiter as a left outer join tMitarbeiter as b
    on a.Vorgesetzter = b.PersNr

select tKunden.vorname, tKunden.nachname from tKunden
  union
select tMitarbeiter.vorname, tMitarbeiter.nachname from tMitarbeiter
order by nachname

select artnr, artBez, Einzelpreis,
  (select avg(Einzelpreis) from tArtikel) as Durchschnittspreis,
  Einzelpreis-(select avg(Einzelpreis) from tArtikel) as Unterschied
from tArtikel

/* Alle Kunden- und Auftragsnummern, die mehr als 20 Stück des Artikels
mit der Artikelnummer 2 bestellt haben */
select AuftrNr, KdNr
from tAuftrag
where 20 <
  (select anzahl from tAuftragsdetails
   where tAuftrag.AuftrNr = tAuftragsdetails.AuftragsNr
   and tAuftragsdetails.ArtNr = 2)

/* alle Artikel, deren Einzelpreis größer als der Durchschnittspreis ist
*/
select ArtNr, ArtBez, Einzelpreis
from tArtikel as Art1
where Art1.Einzelpreis >
  (select avg(Art2.Einzelpreis) from tArtikel as Art2)
```

2.9 Einfügen, Ändern und Löschen von Daten

Einfügen einzelner Datenzeilen:

```
use verkauf
insert dbo.tKunden
values (815,'Helmut','Keil')
go

/*
Fügt neuen Datensatz ein
Version: 1.5
```

```
Datum: 08.05.2007
*/
use verkauf
insert dbo.tKunden
       (Nachname,Vorname,KdNr) – Reihenfolge wird festgelegt
values ('Moser','Stefan',1201)
go

insert dbo.tArtikel
values ('Socken schwarz',1.5); – Identity-Werte nicht explizit festlegen

insert dbo.tArtikel (ArtBez,Einzelpreis)
values ('T-Shirt rot',13.62);

insert dbo.tArtikel (ArtBez,Einzelpreis)
values ('Socken blau',1.21);
```

Einfügen mehrerer Datenzeilen, die aus anderen Tabellen selektiert werden:

```
use Auftrag
insert tKunden   –insert into tKunden
  select PersNr, Vorname, Nachname
  from tMitarbeiter
go

select vorname, nachname into #TempKunden from tKunden

select * from #TempKunden

update tArtikel
  set ArtBez='Hose'
  where ArtNr=2;
```

2.10 Arbeiten mit vordefinierten Funktionen

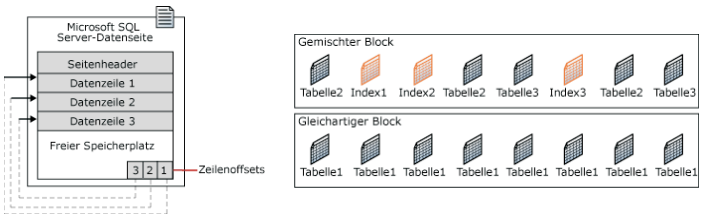
Beispiel: Es soll der in der Spalte ContactName der Tabelle Customers der Datenbank Northwind befindliche Text in Vor- und Nachname geteilt werden.

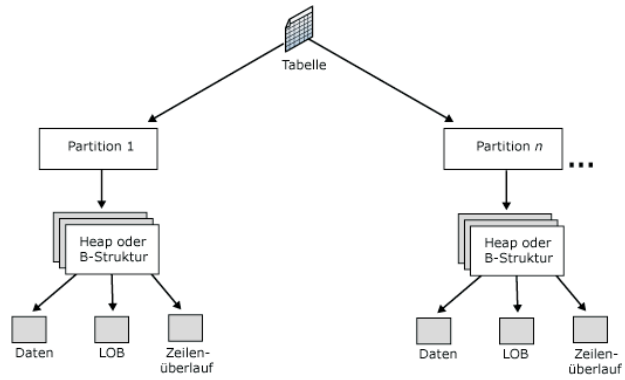
```
select * into Customers2 FROM Customers

– Findet die Länge des Nachnamens
SELECT ContactName, PATINDEX('% %',REVERSE(ContactName)) - 1 FROM
Customers

– Schneidet den Nachnamen aus
SELECT ContactName, Nachname = RIGHT(ContactName,PATINDEX('%
%',REVERSE(ContactName)) - 1) FROM Customers

– Schneidet den Teil vor dem Nachnamen aus
SELECT ContactName, Vorname =
RTRIM(SUBSTRING(ContactName,1,LEN(ContactName) - PATINDEX('%
%',REVERSE(ContactName)))) FROM Customers

– Liefert eine Zahl, wenn im Rest noch ein Blank vorkommt
SELECT ContactName, Nacharbeit = PATINDEX('%
%',RTRIM(SUBSTRING(ContactName,1,LEN(ContactName) - PATINDEX('%
%',REVERSE(ContactName))))) FROM Customers

UPDATE Customers2
SET
  LastName    = RIGHT(ContactName,PATINDEX('% %',REVERSE(ContactName)) -
1),
  FirstName     = RTRIM(SUBSTRING(ContactName,1,LEN(ContactName) -
PATINDEX('% %',REVERSE(ContactName)))) ,
  BearbeitungsPosition = PATINDEX('%
%',RTRIM(SUBSTRING(ContactName,1,LEN(ContactName) - PATINDEX('%
%',REVERSE(ContactName)))))


select ContactName,FirstName,LastName,BearbeitungsPosition, * from
Customers2 where BearbeitungsPosition > 0
```

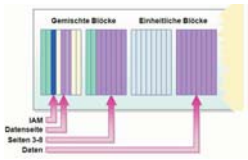
Funktion (Syntax)	Bedeutung
`Len(String)`	Berechnet die Anzahl der Zeichen eines Strings.
`LTrim(String)`	Entfernt führende Leerzeichen in einem String
`RTrim(String)`	Entfernt Leerzeichen am Ende eines Strings
`PatIndex(Muster, String)`	Ermittelt die Position, an der ein Muster das erste Mal im angegebenen String auftritt
`Reverse(String)`	Kehrt einen Text zeichenweise um (aus 'nebel' wird 'leben')
`SubString(String,Beginn,Anzahl)`	Schneidet einen Teil des Strings heraus, beginnend vom Zeichen `"Beginn"` werden `"Anzahl"` Zeichen herausgeschnitten
`Left(String,Anzahl)`	Liefert eine Anzahl von Zeichen, von links beginnend, von einem String
`Right(String,Anzahl)`	Liefert eine Anzahl von Zeichen, von rechts beginnend, von einem String

2.11 Indizes

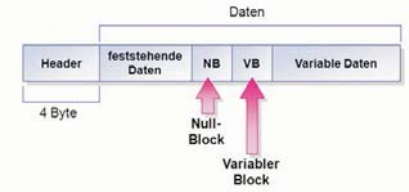
Interne Datenorganisation von SQL Server:



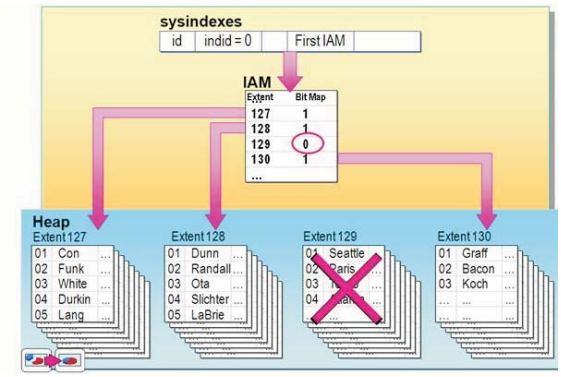




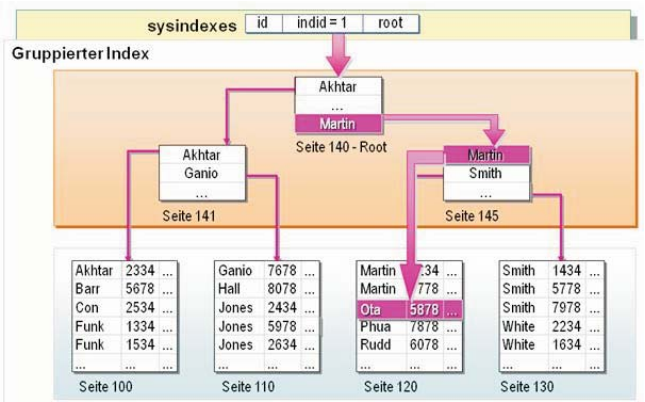
Eine Datenseite ist wie folgt aufgebaut:



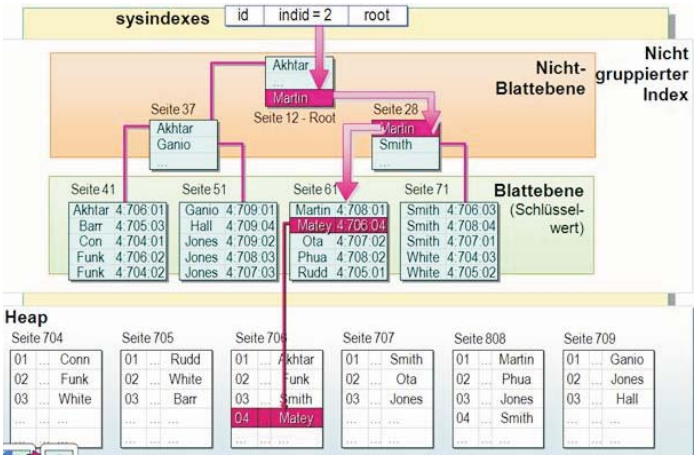
Suchen in einer Tabelle ohne Indizes (=Heap) (Wird auch als *"table scan"* bezeichnet):



Suchen mit gruppiertem Index:



Suchen mit nicht gruppierten Indizes:



Neu ist die Möglichkeit, während der Indexerstellung die Tabellen online zu halten.

```
CREATE INDEX ix_Employee_ManagerID on HumanResources.Employee(ManagerID)
WITH (ONLINE=ON,MAXDOP=1)
```

„*Covered Query*": Abfrage, bei der alle Spalten Teil eines Index sind.

Mit der INCLUDE-Funktion können nun auch Spalten aufgenommen werden, die nicht Teil des Indexes sind:

```
CREATE INDEX ix_AddressDetails on Contact.Address (AddressID)
INCLUDE (AddressLine1, AddressLine2)
```

2.12 Einschränkungen (Constraints)

Spalten- und Tabelleneinschränkungen

Einschränkungen können Spalten- oder Tabelleneinschränkungen sein:

- Eine Spalteneinschränkung wird als Teil einer Spaltendefinition angegeben und gilt nur für diese Spalte.
- Eine Tabelleneinschränkung wird unabhängig von einer Spaltendefinition deklariert und kann für mehr als eine Spalte in einer Tabelle gelten. Tabelleneinschränkungen müssen verwendet werden, wenn mehr als eine Spalte in eine Einschränkung eingeschlossen werden muss.

Primary Key-Einschränkungen, Default-Einschränkungen

```
use Auftrag;
create table dbo.tArtikel
( ArtNr int identity(1,1),
  ArtBez nvarchar(50) NOT NULL,
  Einzelpreis money NOT NULL constraint DF_tArtikel_Einzelpreis default
(0.0),
  constraint PK_tArtikel_ArtNr primary key nonclustered
    (ArtNr ASC) with (ignore_dup_key = off)
);
```

Wenn in einer Tabelle z. B. zwei oder mehr Spalten für den Primärschlüssel verwendet werden, müssen Sie eine Tabelleneinschränkung verwenden, um beide Spalten in den Primärschlüssel einzuschließen. Stellen Sie sich eine Tabelle vor, die Ereignisse aufzeichnet, die für einen Computer in einer Fabrik eintreten. Nehmen Sie weiterhin an, dass unterschiedliche Ereignistypen gleichzeitig eintreten können, dass jedoch nie zwei Ereignisse desselben Typs gleichzeitig eintreten. Dieser Sachverhalt kann in der Tabelle erzwungen werden, indem Sie die type- und die time-Spalte in einen Primärschlüssel einschließen, der zwei Spalten umfasst.

```
CREATE TABLE factory_process
   (event_type   int,
   event_time   datetime,
   event_site   char(50),
   event_desc   char(1024),
CONSTRAINT event_key PRIMARY KEY (event_type, event_time) )
CREATE TABLE tPLZ
(
PLZ char(5) not NULL,
Ort varchar(50) not NULL
CONSTRAINT PK_tPLZ PRIMARY KEY (PLZ, Ort)
);
```

Foreign Key-Constraints

```
alter table dbo.tAuftrag
  add MitarbeiterNr int null;

alter table dbo.tAuftrag
  with check  – vorhandene Datensätze werden überprüft
  add constraint FK_MitarbeiterNr_tMitarbeiter
    foreign key(MitarbeiterNr)
    references dbo.tMitarbeiter(MitarbeiterNr);
```

```
alter table dbo.tAuftrag   – beginnen Sie immer mit der Detailtabelle,
                           – das ist die Tabelle, die den Fremdschlüssel
enthält
  with check               – vorhandene Datensätze werden überprüft
  add constraint FK_KdNr_tKunden
    foreign key(KdNr)      – Fremdschlüssel
    references dbo.tKunden(KdNr)  – Bezug auf Primärschlüssel der anderen
Tabelle
    on update cascade;     – Kaskadierungsoptionen
```

3 Sichten (Views)

Man sollte nie direkt mit den Tabellen, sondern immer mit Abfragen arbeiten.

```
use Auftrag
go

create view dbo.Auftragssicht
as
select
  tAuftragsdetails.AuftrNr,
  tAuftrag.Datum,
  tAuftrag.KdNr,
  tKunden.Vorname,
  tKunden.Nachname,
  tAuftragsdetails.ArtNr,
  tArtikel.ArtBez,
  tAuftragsdetails.Anzahl,
  tArtikel.Einzelpreis,
  Anzahl * Einzelpreis AS Zeilenpreis
from
  tKunden as k inner join tAuftrag as a
    on k.KdNr = a.KdNr
  inner join tAuftragsdetails  as d
    on a.AuftrNr = d.AuftragsNr
  inner join tArtikel as art
    on d.ArtNr = art.ArtNr
where
  d.AuftrNr = 1;
```

WICHTIG: Niemals verknüpfte Primärschlüsselfelder in der Abfrage verwenden! Verknüpfte Felder in der Detailtabelle MÜSSEN in der Abfrage enthalten sein!

4 Gespeicherte Prozeduren (Stored Procedures)

Grundsätzliche Syntax:

```
create proc prKunden
as
select * from tKunden
```

Gespeicherte Prozeduren mit Eingabeparametern

Im folgenden Beispiel wird mit Hilfe eines Eingabeparameters eine Parameterabfrage realisiert:

```
create proc dbo.pSucheKdNr
  @KdNr int
as
  select
    dbo.tKunden.KdNr,
    dbo.tKunden.Vorname,
    dbo.tKunden.Nachname
  from
    dbo.tKunden
  where dbo.tKunden.KdNr = @KdNr;

exec pSucheKdNr 210

exec pSucheKdNr @KdNr=88
```

Verwendung von Rückgabewerten *(return values)*

```
alter proc dbo.pKundeEinfuegen
  @KdNr int,
  @Vorname nvarchar(50),
  @Nachname nvarchar(50)
as
  if (exists (select dbo.tKunden.KdNr from dbo.tKunden
              where dbo.tKunden.KdNr = @KdNr))
  begin
    return -1   –Prozedur wird abgebrochen, Rückgabewert von -1
  end
  insert dbo.tKunden
  values (@KdNr, @Vorname, @Nachname)
  return 0

– Beispiel für Rückgabewert: existiert der Kunde -> -1, sonst 0
declare @ret int
exec @ret = pKundeEinfuegen 37,'Matthias','Gruber'
select @ret
```

Fehlerbehandlung mit TRY-CATCH-Strukturen

Konzept: Der TSQL-Code innerhalb der TRY-Anweisung wird testweise ausgeführt. Tritt ein Fehler auf, so wird sofort zum CATCH-Block verzweigt und die dort angeführten Anweisungen ausgeführt.

Wichtig: Im CATCH-Block muss standardmäßig unbedingt ein ROLLBACK TRAN durchgeführt werden, sonst bleibt die Transaktion im Fehlerfall "hängen".

Funktion	Bedeutung
ERROR_NUMBER()	Fehlernummer
ERROR_LINE()	Zeilennummer, in der der Fehler aufgetreten ist
ERROR_PROCEDURE()	Name der gespeicherten Prozedur, in der der Fehler aufgetreten ist
ERROR_SEVERITY()	Schweregrad: 0-10 … Informationsmeldungen, 11-15 … benutzerdefinierbare Fehler, 16-20 … schwere Fehler, 21-25 … kritische Fehler
ERROR_MESSAGE()	Fehlermeldung
ERROR_STATE()	Statuswert des Fehlers (normalerweise immer 0; sollte derselbe Fehler an mehreren Stellen des Programms auftreten können, kann dies über den Statuswert mitgeteilt werden)

```
create proc dbo.InsertAuftragsdetails
  @AuftrNr int,
  @ArtNr int,
  @Anzahl int
  as
  begin try   – wir versuchen folgenden TSQL-Code
    begin tran
      insert dbo.tAuftragsdetails
      values (@AuftrNr, @ArtNr, @Anzahl)
    commit tran
  end try

  begin catch  – wenn obiger Code fehlerhaft ausgeführt
    rollback tran
    select ERROR_NUMBER() as Fehlernummer,
           ERROR_MESSAGE() as Fehlermeldung
  end catch

– Testfälle:
exec InsertAuftragsdetails 1,99,3
  – Ergebnis: Fehler 547 / Beziehung mit tArtikel verletzt
exec InsertAuftragsdetails 2,2,3
  – Ergebnis: Fehler 2627 / Primärschlüssel bereits vorhanden
exec InsertAuftragsdetails 1,3,NULL
  – Ergebnis: Fehler 515 / NULL-Wert in Anzahl-Spalte verboten
```

Weiteres Beispiel

```
create proc pKundeInsert
  @KdNr int,
  @Vorname nvarchar(50),
  @Nachname nvarchar(50)
as
  begin try
    insert tKunden
    values(@KdNr,@Vorname,@Nachname)
  end try
  begin catch
    select ERROR_NUMBER() Fehlernummer, ERROR_MESSAGE() Fehlermeldung
  end catch
–test
exec pKundeInsert 123,'Max','Muster'
(1 Zeile(n) betroffen)
exec pKundeInsert 123,'Maria','Muster'
(1 Zeile(n) betroffen)
```

Gespeicherte Prozeduren mit Ausgabeparametern

Ausgabeparameter haben den Vorteil, dass sie im aufrufenden Programm weiterverwendet werden können. Während eine gespeicherte Prozedur nur genau einen Rückgabewert haben kann, können beliebig viele Ausgabeparameter verwendet werden.

```
/* Insert in die tArtikel-Tabelle
   Hinweis: ArtBez ist IDENTITY und Primärschlüssel!!
*/
alter PROCEDURE dbo.InsertArtikel
  @ArtBez nvarchar(50)=NULL,
  @Einzelpreis money=NULL,
  @NeueArtNr int OUTPUT
AS
  SET NOCOUNT ON
  if (isnull(@ArtBez,'')='') or
    (isnull(@Einzelpreis,'')='')
  begin
    raiserror(50011,1,16)
    return
  end
  insert tArtikel
    (ArtBez, Einzelpreis)
  values
      (@ArtBez, @Einzelpreis)

  SET NOCOUNT OFF
  – SELECT @NeueArtNr = @@IDENTITY
    set @NeueArtNr = SCOPE_IDENTITY()
GO
```

Benutzerdefinierte Fehlermeldungen müssen definiert werden; dazu steht die gespeicherte Systemprozedur sp_addmessage zur Verfügung:

```
/* Hinzufügen neuer benutzerdefinierter Fehlermeldung */
exec sp_addmessage 50011, 16,
   'Datensatz nicht hinzugefügt, da ArtBez NULL ist','us_english'

/* Test der Stored Procedure */
declare @neuenummer int
exec insertArtikel 'Schuhe blau',12.45, @neuenummer OUTPUT
select @neuenummer
go

/* Test der Stored Procedure: Fehlermeldung */
declare @neuenummer int
exec insertArtikel '',12.45, @neuenummer OUTPUT
go
```

Funktionen

```
CREATE FUNCTION fn_HoleOrt (@plz varchar(10))
  RETURNS varchar(50)
AS
BEGIN
  return (SELECT PoOrt FROM tPlzOrt WHERE PoPlz = @plz)
END
GO
```

5 Trigger

Trigger sind mit gespeicherten Prozeduren vergleichbar, die auf Grund einer Datenbankaktion automatisch ausgeführt werden.

Man unterscheidet:

- **DML-Trigger** (seit SQL Server 7.0 möglich): werden durch eine INSERT-, UPDATE- oder DELETE-Aktion ausgelöst
- **DDL-Trigger** (ab SQL Server 2005): werden durch DDL-Statements wie CREATE, ALTER, DROP ausgelöst

Bei den DML-Triggern unterscheidet man weiter:

- AFTER-Trigger: werden **nach** einem INSERT, UPDATE oder DELETE ausgeführt
- INSTEAD OF-Trigger: werden **statt** eines INSERT, UPDATE oder DELETE ausgeführt

```
create trigger trNeuerKunde on tKunden
  after insert as
begin
  set nocount on
  insert dbo.tProtokoll
    select getdate(), user_name(), 'insert',inserted.KdNr
    from inserted
  set nocount off
end
```

Die logische **inserted**-Tabelle enthält die einzufügenden bzw. eingefügten Datensätze. Analog dazu gibt es auch eine logische **deleted**-Tabelle, die gelöschte Datensätze enthält.

```
/* Testen des Triggers */
insert tAuftrag (KdNr, Datum, MitarbeiterNr)
values (109, '24.05.2007',220);
```

Beispiele

```
alter trigger trAendernKunde on tKunden
  after update as
begin
  set nocount on
  insert dbo.tProtokoll
    select getdate(), user_name(), 'update',inserted.KdNr
    from inserted
  set nocount off
end

create trigger trLoeschenKunde on tKunden
  after delete as
begin
  set nocount on
  insert dbo.tProtokoll
    select getdate(), user_name(), 'delete',deleted.KdNr
    from deleted
  set nocount off
end
```

Windows Workflow Foundation

Thomas Reinwart

1. Workflow -Einführung

Warum einen Workflow verwenden

Ein Workflow ist ein Modell bestehend aus Aktivitäten das einen Prozess beschreibt. Diese Aktivitäten koordinieren Personen oder Maschinen.

Mit der Einbindung eines Workflows gibt es in der Architektur einer Applikation neue Möglichkeiten, die *Business Logic (BL)* zu definieren. Statt wie bisher unüberschaubaren Code mit vielen `if else`-Zweigen zu kodieren, gibt es nun die Möglichkeit, dies in einen Workflow zu verpacken. So lässt sich nun die BL mittels eines graphischen Designers erstellen, damit ist dies wesentlich flexibler bei Veränderungen. Zudem ist es übersichtlich dokumentiert. Das betrifft auch weit komplexere Prozesse eines Unternehmens. Mit einem Workflow lassen sich Abläufe besser koordinieren.

Windows Workflow Foundation (WF)

Der Vorteil, einen Workflow zu verwenden ist, dass damit der Prozess recherchiert, beschrieben und daraus ein Modell erstellt wird. Meist werden die Prozesse als *UML*, *use cases* oder *flow chart* dokumentiert. Beim Verwenden von WF sind die Beschreibung und der Workflow ein und das selbe System, d.h. es gibt eine visualisierte Darstellung des Workflow beim Design und zur Laufzeit, der Code der Activities ist mit dem Workflow verbunden. Somit gibt es keine getrennte Dokumentation die auseinander laufen kann.

Welcher meiner laufenden Workflows derzeit in welchem Status sich in welchem Step befindet, zeigt mir Tracking im selben Design visuell an.

Ein Workflow kann im Designer verändert werden, ohne dass dabei Code geändert werden muss. Ich kann einen Workflow in ein Produkt einbauen, dabei wird jeder Workflow für den Kunden angepasst, der Rest bleibt gleich. Bzw. der Kunde passt sich seinen Workflow selber an.

Meist läuft am Rechner eine Vielzahl von Programmen, die nur von Zeit zu Zeit echte Dienste leisten. Die meiste Zeit jedoch verbringen diese Anwendungen damit, auf Usereingaben oder auf ein anderes programatisches Event zu warten. Die Anwendungen laufen also und tun nichts anderes als Ressourcen zu verbrauchen.

In einem Workflow können die Objekte serialisiert und derserialisiert werden, die Persistierung in die Datenbank. Somit ist der Status, in dem sich der Workflow der Anwendung befindet eingefroren. Wenn nun ein bestimmtes Event eintritt, beginn der Workflow wieder zu arbeiten. Durch diese persistieren des Zustandes ist es möglich, das Laden auf einen anderen Rechner und Zeitpunkt durchzuführen. Man hat damit selber keinen Implementierungsaufwand, sich um die Persistierung eines Workflows Status zu kümmern.

Ein Workflow kann in jedem Projekttyp eingebunden werden, vom simplen Comand Line Tool bis Windows Forms, XAML, Asp.net. Innerhalb des Workflows kann über Code Activities wiederrum eigene Assemblies aufgerufen werden, oder auch WebServices eingebunden werden.

Komponenten

- *Base Activity Library*: mit der WF mitgelieferte Activities
- *Runtime engine*
- *Runtime services*: Hosting und Kommunikation
- *WF Visual Designer*: Control kann auch in der eigener Applikation gehostet werden

1.1 Sequential oder State Workflows verwenden

Die Windows Workflow Foundation bietet zwei Workflow Arten an, den *Sequential* und den *State Workflow*. Beim Projektbeginn muss ich mir im Klaren sein, welcher Workflow besser zu meinem Projekt passt.

1.1.1 Der State Workflow

Der Workflow befindet sich in einem bestimmten Status und wartet auf Events um in einen anderen Status zu gelangen. Der neue Status kann irgendein anderer Status sein. Status können verschachtelt sein. Gut geeignet für menschliche Interaktion.

Dieser Workflow führt die Aktivitäten aufgrund des Status aus. Der Workflow hängt stark von den externen Modulen ab. Userinteraktion bzw. Ergebnis der externen Module beeinflussen den Status. Zustandsübergänge werden durch externe Events ausgelöst.

Er eignet sich dann, wenn mehr Useraktion gewünscht wird. Wenn dieser Workflow eine Benachrichtigung an einen anderen Benutzer oder ein anderes System sendet, gibt es Antworten in Form von Events. Dadurch wird ein Prozess Status durch einen anderen Prozess Status getauscht.

Ein Bespiel für einen State Workflow ist das Zusammenspiel von verschiedenen Interaktionen einer Produktion eine Maschine mit einem Bediener.

Start – Material entnehmen – Material reinigen – Material bearbeiten – Material Kontrolle – Material bearbeiten – Material Ausschuss oder ***OK***

Dazwischen kann jeder einzelne Step auch wieder zum Vorgänger springen. Der Workflow wird hier lange Zeit in einem der Steps verbringen.

Der Unterschied zum *Sequential Workflow* ist, dass beim *State Workflow* auch ein vorheriger Status erreicht werden kann.

1.1.2 Der Sequential Workflow

Der Plan eines sequentiellen Workflows ist vorgegeben. Er kann Bedingungen, Schleifen usw. beinhalten. Dies passt mehr zu automatisierten Prozessen oder starren Protokollen. Er hat einen definierten Start und ein definiertes Ende, kann aber auch ewig laufen.

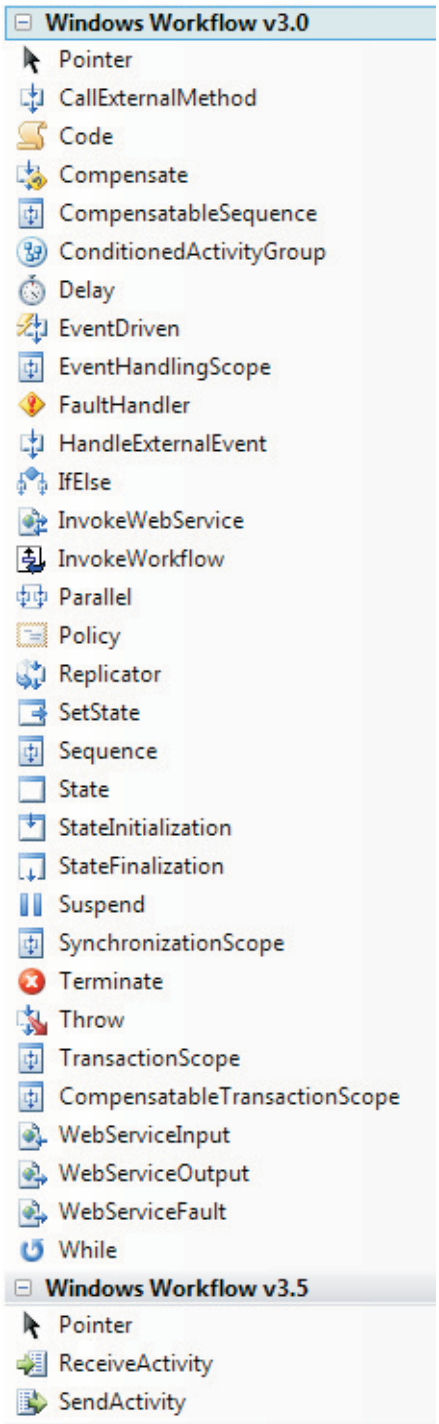
Dieser Workflow führt die Activities hintereinander aus. Der Workflow hat dabei ausführbaren Tasks unter Kontrolle. Es kann Userinteraktion geben, der Workflow kann aber auch vollautomatisch laufen.

Er ist ideal für die Umsetzung von Business Prozessen, die komplex sind.

Dieser Workflow arbeitet eine Activity nach der anderen ab, es sei denn eine Exception wird während aber Workflow Abarbeitung gefeuert oder wenn eine TerminateActivity ausgeführt wird.

Beispiel: Daten von einer Quelle lesen, Daten verarbeiten, eine Benachrichtigung schicken, Ergebnisdaten schreiben. Die Userinteraktion ist also nicht unbedingt notwendig.

Zur Verfügung stehende Activities im Workflow



Activity: alles in einem Workflow ist eine Activity, auch der Workflow selber, der einen bestimmten Typ einer Activity darstellt.

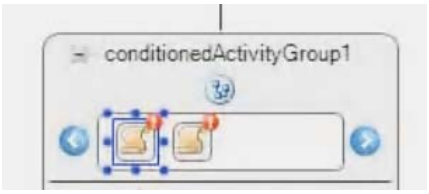
CallExternalMethod: Damit lassen sich Methoden außerhalb eines Workflows aufrufen. InterfaceType und MethodName müssen dabei definiert werden. InterfaceType gibt an, welches Runtime Service genutzt werden soll, wenn einen Activity ausgeführt wird. MethodName gibt an, welche Methode des angegebenen Interface aufgerufen werden soll.

Code: Hiermit lässt sich eigener .net Code im Workflow ausführen. Der Sourcecode ist getrennt vom Workflow.

Compensate: Ist eine Error Activity und kann zu einer Exception Handler Activity hinzugefügt werden. Kann genutzt werden um ein Rollback der Änderungen im Falle eines Fehlers durchzuführen.

Compensatable Sequence: Kann für einen bestimmten Typ eines Fehlers verwendet werden. Funktioniert ähnlich wie mit Datenbank Transaktionen. Der Unterschied ist, das die Datenbank solange sperrt bis alle Updates fertig sind, also für eine kurze Zeit. Eine Compensate Action kann bei Transaktionen verwendet werden, die lange andauern, um die original Transaktion rückgängig zu machen.

Conditioned Activity Group: Dies ist eine bedingte Aktivität, die andere Aktivitäten, basierend auf Bedingungen, die für die Gruppe gelten, oder die Aktivitäten die der Gruppe zugeordnet, ausführt.



Delay: Bei der Verwendung einer DelayActivity wird der Wartezeitraum und die Methode angegeben, die im Fall der Zeitüberschreitung ausgeführt wird. Im Beispiel eine Genehmigungsworkflows kann man ein Delay verwenden. Wenn nun eine Person innerhalb des angegeben Zeitraums nicht reagiert, wird der Alternativzweig im Workflow (z.B. andere Person kontaktieren) verwendet.

Event Driven:Diese Activity kann andere Activities beinhalten, die ausgeführt werden wenn dieses Event eintritt.

Event Handling Scope*: kann eine Anzahl von Activities beinhalten. Während diese Activities ausgeführt werden, können die Events empfangen werden.*

Fault Handler: Eine Error handling Activity, die wie ein catch block funktioniert.

Handle External Event: Die HandleExternalEventActivity wird in Verbindung mit der CallExternalMethodActivity für die In und Out Kommunikation mit einem lokalen Dienst verwendet. Sie können diese Aktivitäten direkt für die allgemeine Kommunikation verwenden. Oder Sie können Unterklassen von HandleExternalEventActivity und CallExternalMethodActivity erstellen um Aktivitäten, die streng an bestimmte Ereignisse und Methoden an ein Interface gebunden ist das ein ExternalDataExchangeAttribute besitzt, zu erzeugen.

If Else: Kann man sich als if else so wie in der Programmiersprache vorstellen. Über eine Condition wird die Bedingung festgelegt.

Invoke Web Service: kann ein Methode eines WebService mittels Proxyklasse ausführen. Parameter können übergeben und Empfangen werden.

Invoke Workflow: Ruft den zugeordneten Workflow auf Parallel: Im Workflow werden Activities als single thread ausgeführt. Es ist also keine wirkliche parallele Ausführung. Die Workflow runtime verarbeitet die Activity Queue pro Workflow Instanz im Prinzip von FIFO (first in, first out) ab.

Policy: Dies stellt eine Auflistung von Regeln dar. Eine Regel hat eine Bedingung und Aktion(en), die durchgeführt werden, wenn die Bedingung erfüllt ist. Das erlaubt einen regelbasierten Workflow, anstatt einem IfElse basierendem Workflow.

Replicator: Ist auch eine Bedingungsaktivität. Wie wenn man eine "For Each" Anweisung verwendet. Zur Laufzeit werden eine Anzahl von Instanzen einer Aktivität erstellt, die abgeschlossen werden müssen, bevor die Replikator Aktivität beendet wird.

Set State Activity: Dies ist ein flow activity, die für den Statuswechsel bei einem State Machine Workflow verwendet wird.

State Activity: Ist eine flow activity und stellt einen Status in einem State Machine Workflow dar.

State Initialization Activity: Diese Aktivität ist Bestandteil einer State Aktivität, die aus anderen Aktivitäten besteht, die ausgeführt werden, wenn die State Aktivität initialisiert wird.

Sequence: Ist eine zusammengesetzte Aktivität, die mehrere sequenzielle Aktivitäten beinhaltet. Es bietet eine einfache Möglichkeit, diese Tätigkeiten zu verknüpfen, die damit in Folge ausgeführt werden.

Suspend: Ist eine flow-Aktivität und kann den Ablauf eines Workflows für ein Einschreiten pausieren, wenn eine Fehlerbedingung (error condition) eintritt.

Synchronization Scope: Synchronisiert Workflow Aktivitäten

Terminate: Diese Aktivität beendet die Ausführung des Workflows sofort, wenn ein Fehler auftritt. Sie protokolliert auch den Fehler.

Throw: Mit dieser Aktivität können Sie eine Exception auslösen.

Transaction Scope: Diese Aktivität bietet Transaktionsunterstützung. In dieser Aktivität können sich Aktivitäten befinden, die eine Transaktion bilden.

Compensatable Transaction Scope: Es gibt zwei Aktivitäten die dieses Interface implementieren: CompensatableTransactionScopeActivity und CompensatableSequenceActivity.

WebService Input, Output, Fault: WF WebService Integration

While: Dies ist eine bedingte Aktivität und ist wie eine `"while"` Bedingung in der Anwendung. Es wird eine andere Aktivität ausgeführt, bis eine Bedingung erfüllt ist. Die Bedingung kann ein Code oder einfach eine Regel basierende Bedingung sein.

Condition: es gibt die CodeCondition und die RuleConditionReference

CodeCondition: diese führt eine Methode der eigene code behind Klasse aus. Diese gibt `true` oder `false` zurück. Mit Code Condition sind Entwickler vertraut, hiermit lassen sich komplexe Bedingungen einfacher als in einer RuleConditionReference abbilden. Das Ergebnis einer CodeCondtion wird als Result Property der ConditionalEventArgs zurückgeliefert.

RuleConditionReference: diese kann extern als XML gespeichert werden, die Extension ist `.rules`. Das hat den Vorteil, dass dies extern durch andere Tools geändert werden kann, auch zur Laufzeit.

Parameterübergabe in Workflows: Bei einer Activity gibt es normalerweise einen Zusammenhang zu den Bewegungsdaten, also zum Bespiel zur Customer Id. Zum Unterschied zum Aufruf von Klassen in .net, wo Parameter bei Methoden übergeben werden, gibt es bei der Parameterübergabe in Workflows ein Dictionary von Name/Values.

1.1.3 Für welchen Workflow soll ich mich entscheiden?

Ein State Workflow wartet auf externe Events bevor er in einen anderen Step springt.

Ein Sequential Workflow ist ein zusammenhängender Fluss von Activities. Activities in einem Sequential Workflow warten nicht auf externe Instanzen um dann in den nächsten Step zu springen.

Man kann jeden State Workflow auch als Sequential Workflow abbilden. Früher order später wird dieser aber unüberschaubar werden.

2. Workflow Speicherformat

XOML *(Extensible Object Markup Language)* ist eine auf XML basierende Syntax. Ein Workflow kann als XOML gespeichert werden.

2.1 SqlPersistenceService

Per default laufen alle Workflows im Memory, solange auch die Applikation läuft, die den Workflow hostet. Zum Persistieren jeder einzelner Instanz des laufenden Workflows kann das mitgelieferte SqlPersistanceService verwendet werden. Damit wird der Zwischenstand der Workflows gespeichert, die Host Applikation kann jederzeit beendet und neu gestartet werden, außerdem können Workflows aus dem Speicher entladen werden wenn sie idle gehen. (Spart Memory, Performance)

Der Workflow kann damit am Filesystem oder in der Datenbank (SQL Server, SQL Express, …) gespeichert werden.

```
// Add the SqlWorkflowPersistenceService service
WorkflowPersistenceService persistenceService =
  new SqlWorkflowPersistenceService(
    "Initial Catalog=SqlPersistenceService;" +
    "Data Source=localhost;" +
    "Integrated Security=SSPI;",
    true,
    TimeSpan.FromHours(1.0),
    TimeSpan.FromSeconds(5.0));
workflowRuntime.AddService(persistenceService);
```

Parameter

ConnectionString: Verbindung zur bestehenden Persist DB

UnloadOnIdle: muss auf true stehen, sonst wird nichts persistiert. Gibt an, dass der Workflow entladen wird, wenn er idle ist. Default Wert false.

LoadingInterval: Zeitintervall, in der die Persist DB gepollt wird, um die persist records zu schreiben.

DB Installationsscript:

`c:\WINDOWS\Microsoft.NET\Framework\v3.0\Windows Workflow Foundation\SQL\EN\`

Tipp: SqlPersistanzService DB und Tracking DB können einzeln und getrennt verwendet werden. Wenn beides verwendet wird, empfiehlt sich aber beides in eine DB (SQL Server 2000, SQL Server 2005, oder SQL Server 2005 Express Edition) zu geben.

Microsoft Distributed Transaction Coordinator Service muss gestartet werden. Bei Verwendung eines Remote Servers muss Port 135 für Microsoft Distributed Transaction Center (MSDTC) offen sein.

2.2 Tracking

Da Workflows im Hintergrund laufen und die Workflow an sich auch komplexe Ausmaße annehmen werden, können diese mittels Tracking visualisiert werden. Bei der Instanziierung des Workflows wird das SqlTracking Service angegeben. Somit wird jede Instanz der Workflows getrackt.

Das Tracking ist für laufende als auch für beendete Workflows möglich, d.h. ich kann mir während des Ausführens als auch im Nachhinein den durchlaufenen Workflow ansehen. Die Darstellung erfolgt graphisch, der durchlaufene Workflow wird abgehackt dargestellt, der aktuellen Step mit einem grünen Dreieck.

DB Installationsscript liegt im Installationsverzeichnis:

`c:\WINDOWS\Microsoft.NET\Framework\v3.0\Windows Workflow Foundation\SQL\EN\`

3. Praxis

3.1 Workflow Hosting

Beispielanwendungen

- Automatisierungen von Abläufen
- Dokumenten Management
- Ticket Systeme
- Seitennavigation
- Komplexe Business Logic
- Häufige Änderungen an der Business Logic

Das Hosting kann in einem Service, über Webservice, asp.net, … erfolgen.

In diesem Beispiel in einem Windows Service:

```
#region fields
  WorkflowRuntime _wfruntime = new WorkflowRuntime();
  static AutoResetEvent _waitHandle = new AutoResetEvent(false);
  private System.Collections.ObjectModel.ReadOnlyCollection<WorkflowInstance>
    _loadedWorkflows;
#endregion
protected override void OnStart(string[] args)
{ ThreadPool.QueueUserWorkItem(new WaitCallback(RunWorkflow), args);
  _wfruntime.WorkflowCompleted +=
    new EventHandler<WorkflowCompletedEventArgs>(OnWorkflowCompleted);
}
private void RunWorkflow(object obj) {
  try
  {
    // Load WF from SqlWorkflowPersistenceService // Create the WorkflowRuntime
    using (WorkflowRuntime workflowRuntime = new WorkflowRuntime())
    {
       // Add the SqlWorkflowPersistenceService service
       WorkflowPersistenceService persistenceService =
         new SqlWorkflowPersistenceService(
           "Initial Catalog=SqlPersistenceService;" +
          "Data Source=localhost;Integrated Security=SSPI;",
           true, TimeSpan.FromHours(1.0), TimeSpan.FromSeconds(5.0));
       workflowRuntime.AddService(persistenceService);
       // Tracking
      workflowRuntime.AddService(
        new SqlTrackingService("Initial Catalog=Tracking;Data Source=localhost;" +
          Integrated Security=SSPI;"));
      // Set up the WorkflowRuntime event handlers
      workflowRuntime.WorkflowCompleted += OnWorkflowCompleted;
      workflowRuntime.WorkflowIdled += OnWorkflowIdled;
      workflowRuntime.WorkflowPersisted += OnWorkflowPersisted;
      workflowRuntime.WorkflowUnloaded += OnWorkflowUnloaded;
      workflowRuntime.WorkflowLoaded += OnWorkflowLoaded;
      workflowRuntime.WorkflowTerminated += OnWorkflowTerminated;
      workflowRuntime.WorkflowAborted += OnWorkflowAborted;
      _loadedWorkflows = workflowRuntime.GetLoadedWorkflows();
      Console.WriteLine("Workflows loaded: ", _loadedWorkflows.Count);
    }
  }
  catch (Exception ex)
  {
    EventLog.WriteEntry("Hosting WF in an Windows Service", e.ToString(),
      EventLogEntryType.Information, 1000);
  }
}
```

3.2 Workflow Tracking

Das Tool wird in den Samples mitgeliefert.

Workflow Monitor Sample

http://msdn.microsoft.com/en-us/library/ms741723.aspx

Um dies nutzen zu können, sind folgende Schritte notwendig:

- SQL Tracking DB muss angelegt worden sein.
- Eigener Workflow ist design und compiliert, Workflow wurde gestartet
- Erzeugt Workflow Assembly entweder ins Bin Verzeichnis vom Workflow Monitor kopieren oder die Assembly in den GAC geben (`gacutil –i demo.dll`)
- Workflow Tracking starten, Tracking DB angeben

3.3 Praxis Beispiel

In jedem organisieren Unternehmen gibt es wiederkehrende Abläufe, die man in einem Workflow Prozess abbilden kann. Dies betrifft einfache und komplizierte Vorgänge. Alleine durch die Analyse innerhalb einer Firma ergeben sich oft Schwachstellen, Lücken oder Missverständnisse die Arbeitsvorgänge undurchsichtig, unkontrollierbar und zeitaufwendig gestalten.

Es bietet sich an, dies als elektronischen Workflow abzubilden.

Anhand eines einfachen Beispiels möchte ich dies veranschaulichen.

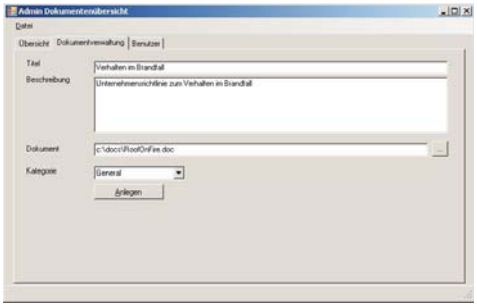
Es handelt sich dabei um einen Genehmigungsprozess, bei dem ein bereitgestelltes Dokument durch einen Gruppe oder einen Benutzer akzeptiert werden muss.

Ablauf

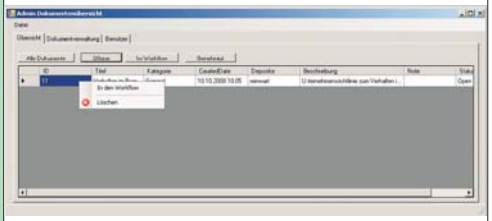
Eine leitende Person erstellte Dokumente und stellt diese in den Workflow. In dem Workflow sind drei Abteilungen definiert: IT User, System User, und allgemeine Benutzer.

Im Workflow sind diese Gruppen durch eine IfElseActivity aufgeteilt, daher kann jede Gruppe eine besondere Businnes Logic (BL) aufweisen. Eine solche BL lässt sich in einer CodeActivity unterbringen. In einer solchen CodeActivity lässt sich eigener Code ausführen.

Die Nutzung des zuvor erstellten Workflows ist in diesem Fall eine simple .net Windows Forms Anwendung, ist aber vom Workflow völlig unabhängig.



Ein Dokument mit den Attributen Titel, Beschreibung und Kategorie wird angelegt.



In der Übersichtsliste erscheint das Dokument und kann in den Workflow übernommen werden.

Technischer Teil

In der Applikation wurde das Persistence Service und das TrackingService der Windows Workflow Foundation eingebunden.

Persistence Service

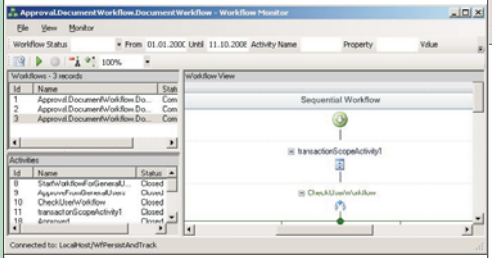
Jede Workflow Änderung eines Status wird in die Datenbank geschrieben.

TrackingService

Bietet die Möglichkeit, bei Instanz eines Workflow den aktuellen Status zu erfassen

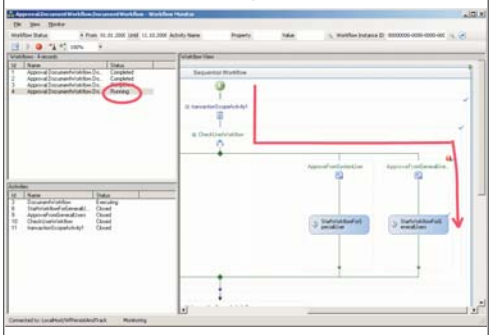
Tracking Monitor

Zeigt den aktuellen Workflow an ….

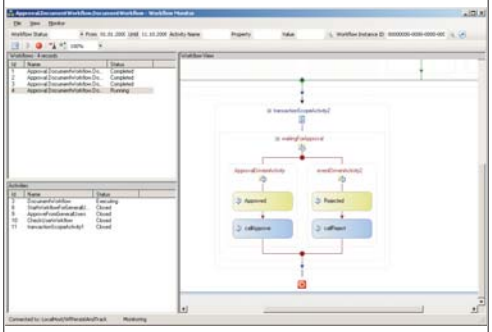


Damit lässt sich jeder aktuelle Step jedes einzelnen Workflow Überwachen und verständlich darstellen.

Das Dokument wurde in der Workflow gestellt, der Workflow läuft. *(workflow running)*

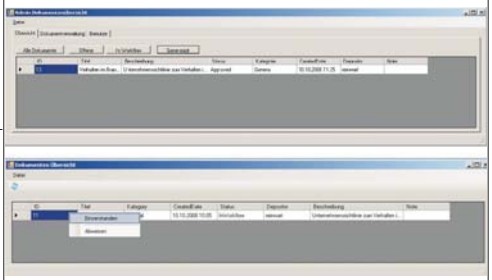


Der Workflow hat nun den Weg für die Kategorie General User genommen und wartet nun auf die Bestätigung durch den zugewiesenen User.



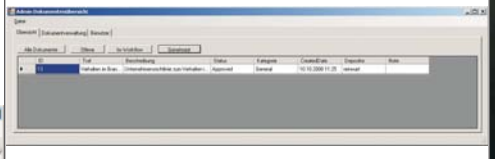
Der User hat nun die Möglichkeit, das Dokument, das nun diesen Workflow durchläuft zu bestätigen oder abzuweisen.

Für den User gibt es eine andere Applikation. Zu Beginn muss er sich authentifizieren, nach der Anmeldung sieht er die für ihn bestimmten Dokumente, für die eine Zustimmung notwendig ist.



Er liest sich den Inhalt des Dokuments durch und ist damit einverstanden.

Die Anwendung aus der Sicht des Administrators: Das Dokument wurde genehmigt und ist als *„Approved"* in der Liste geführt.



Der Administrator hat laufend die Übersicht, welche User die Dokumente bereits akzeptiert haben. Bei einer Überschreitung eines Zeitraumes könnte man in diesem Demo Workflow ohne viel Aufwand auch um eine Eskalationsstufe (z.B. automatische generiertes Email) erweitern.

4. Einschränkungen mit WF 3.0

Performance: Microsoft selbst dokumentiert dies in dem Dokument "Performance Characteristics of Windows Workflow Foundation".

(http://msdn.microsoft.com/en-us/library/aa973808.aspx)

Versionierung von Workflows

Ein Workflow kann jederzeit geändert werden, aber die bestehenden Workflows kann man nicht mit der neuen Definition weiterlaufen lassen. Wenn Sie bei einem Prozess plötzlich am Ende noch einen Schritt brauchen, können Sie diesen also nicht bei Workflows anwenden, die bereits gestartet sind (auch wenn diese langlebig sind).

Keine Kompatibilität zu SSIS-Workflows

Microsoft SQL Server Integration Services (SSIS) bietet eine ähnliche Umgebung wie WF, ist jedoch in keinster Weise kompatibel zu WF. Das Entwicklungsteam von SSIS hat sich aktiv gegen die Verwendung von WF als Basis ausgesprochen (einerseits war SSIS vor WF auf dem Markt, andererseits ist SSIS mehr auf Massendatenverarbeitung fokussiert, dafür wäre WF zu langsam).

Designer Control Hosting

Eigener Code zusätzlich zum WF Control notwendig, für einen WF Beginner nicht einfach. Mit dem Designer Control der Version 4.0 wird es einfacher sein.

5. Zukünftige Version WF 4.0

Auf der PDC 2008 wurde von Kenny Wolf, einem Microsoft Architect von WF und WCF, die Zukunft von WF in der Version 4.0 präsentiert. Workflow 4.0 soll zusammen mit dem .net Framework 4.0 und Visual Studio 2010 auf den Markt kommen.

In der Vorführung erkennt man die neue einfachere Oberfläche für das Design eines Workflow Prozesses. Aufgrund der Erkenntnis, das die WF Versionen 3.0 und 3.5 Schwächen im Bereich Funktionsumfang, Bedienung, Komplexität und Leistung aufweisen und die Kundenwünsche damit nicht immer befriedigt werden können, wurde WF von Grund auf neu geschrieben. Das Konzept der Aktivitäten im Workflow und die Persistierung und die Möglichkeit von Monitoring bleibt. Es soll eine 10 bis 100-fache Leistungssteigerung geben, ebenso eine volle Kontrolle über die Persistierung. WF 4.0 wird rein deklarativ sein.