JET-SQL

für Fortgeschrittene

Karl Donaubauer
www.donkarl.com

5. AEK 28./29.9.2002
Nürnberg

SQL-GESCHICHTE UND -STANDARDS **3**
DEFINITION UND ENTSTEHUNG 3
SQL-86 3
SQL-89 3
SQL-92 3
SQL-99 (SQL3) 4
SCHREIBSTILE UND KONVENTIONEN 4
ACCESS UND JET-SQL **4**
KOMPATIBILITÄT ZU DEN STANDARDS 4
WAS FEHLT IN ACCESS? 4
BESONDERHEITEN BEI DER SYNTAX IN ACCESS 5
WAS BIETET ACCESS AN ERWEITERUNGEN/BESONDERHEITEN? 5
Einsatzorte für SQL *5*
VBA *5*
Erweiterungen des SQL-Standards *5*
AUSWAHLABFRAGEN **6**
BESONDERHEITEN BEI JOINS 6
DISTINCTROW 6
TOP UND TOP PERCENT 7
WITH OWNERACCESS OPTION 7
TRANSFORM (KREUZTABELLENABFRAGE) 7
BESONDERHEITEN BEI UNION 7
ZUGRIFF AUF EXTERNE DATEN 8
eingebundene Tabellen *8*
IN-Klausel *8*
Direkter Bezug *8*
UNTERABFRAGEN 9
Unterabfragen in Spalten-Ausdrücken *9*
Unterabfragen in der WHERE-Klausel *9*
Unterabfragen in der FROM-Klausel *9*
DATA MANIPULATION LANGUAGE (DML) **10**
UPDATE (AKTUALISIERUNGSABFRAGE) 10
SELECT INTO (TABELLENERSTELLUNGSABFRAGE) 10
DELETE (LÖSCHABFRAGE) 10
INSERT INTO (ANFÜGEABFRAGE) 10
HÄUFIGE PROBLEME BEI AKTIONSABFRAGEN 11
DATA DEFINITION LANGUAGE (DDL) **11**
ERLAUBTE DDL-STATEMENTS 11
DATENTYPEN 11
CONSTRAINT 12
SQL 92 ERWEITERUNGEN IN JET 4 ..... **13**
CREATE TABLE UND ALTER TABLE 13
VIEWS UND STORED PROCEDURES 14
TRANSAKTIONEN 15
SICHERHEIT 15
PERFORMANCE **16**
SHOWPLAN 16
RUSHMORE 17
JET–INTERNE VORGÄNGE UND DIE QUERY ENGINE 18
PERFORMANCE-TIPPS 19
QUELLEN **21**
BÜCHER 21
WEBSEITEN 21
ACCESS-ONLINE-HILFE 22

SQL-Geschichte und -Standards

Definition und Entstehung

SQL = Structured Query Language
Aussprache "ess-k(j)u-ell" oder "siquell"

Entstanden Anfang der 1970er in den Forschungslabors von IBM (Team v. Ronald Chamberlin).
SEQUEL (Structured English Query Language)öffentlich vorgestellt 1974 bei einer Konferenz der Association for Computing Machinery (ACM)
1976/77 SEQUEL/2 bei IBM

Name kurz danach aus rechtlichen Gründen in SQL geändert.

Grundlagen von SQL liegen u.a. in der Forschungssprache SQUARE (Specifying Queries As Relational Expressions). 1975 Ziel relationale Algebra in englisch klingenden Sätzen.

1977 Firmengründung "Relational Software", Ziel Entwicklung eines Datenbanksystem auf Basis von SQL **à** 1979 Produkt ORACLE = 1. relationales Datenbank-Managementsystem
ca. 1980 INGRES, dann SQL/Data System (SQL/DS) und Database 2 (DB2) von IBM

Standardisierung durch langjährige Diskussionen in technischen Komitees von ANSI (American National Standards Institute)

SQL-86

erste große Standardisierung, in den meisten Dialekten erfüllt
wesentliche Basis für diese 1. Standardisierung war DB2
von ISO (International Organization for Standardization) 1987 übernommen

SQL-89

einige Erweiterungen des Standards v.a. Referentielle Integrität

Beide Standards unvollständig z.B. keine Befehle zur Änderung der Strukturen wie ALTER oder DROP, wenige Sicherheits-Befehle GRANT noch ohne REVOKE

SQL-92

viele Neuerungen, v.a. Entwicklungen aus den wichtigen kommerziellen DBMS floßen ein
Befehle zur Strukturänderung, Manipulation von Zeichenketten, Datum- und Uhrzeit-Funktionen, Sicherheitsfunktionen

Kein einziges kommerzielles DBMS unterstützt bisher SQL/92 komplett.
Gründe: Komplexität des Standards - schwierige Implementierung, Eigeninteressen der Hersteller

Standard besteht aus 3 Ebenen

1. ENTRY SQL

v.a. Mängel von SQL/89 behoben, am leichtesten zu implementieren

2. INTERMEDIATE SQL
inkludiert ENTRY SQL und den Großteil der neuen Funktionen, unter Einfluss bestehender kommerzieller Systeme

3. FULL SQL
inkludiert Ebenen 1 und 2 und kompliziertere Funktionen

"Erweiterungen" sind Funktionen in DBMS, die über den Standard hinausgehen

SQL-99 (SQL3)

v.a. Objektrelationale Erweiterungen als Neuheit
Standards in komplizierte Einzelteile zerlegt

Schreibstile und Konventionen

- Schlüsselwörter in Großbuchstaben
- jeden Abschnitt in neue Zeile
- Aliase verwenden, mit AS verdeutlichen (oft a, b... oder T1, T2...)
- Qualifizierte Namen (Tabelle.Feld)

z.B. aus
select Vorname, Nachname from tblKunden
wird
SELECT a.Vorname, a.Nachname
FROM tblKunden AS a

Access und JET-SQL

Kompatibilität zu den Standards

- von Anfang an Unterstützung für SQL-89 Level 1
- einige prioritäre Erweiterungen - wie fast alle DBMS.
- ab Jet4 d.h. Access 2000 weitergehende Unterstützung für SQL-92 v.a. über den JET-OLE-DB Provider (d.h. per ADO).

Was fehlt in Access?

- Stapelverarbeitung von SQL-Statements
- Trigger
- Cursor
- Temporäre Tabellen
- Full Outer Join und Union Join
- Case und
- viele weitereSQL-Befehle (bes. vor Jet 4)

Zitat http://support.microsoft.com/default.aspx?scid=kb;en-us;Q92691
ACC: Unsupported SQL Keywords Not Listed in Microsoft Access
Access 1.0 - 97

Microsoft Access documentation lists all SQL keywords that are supported in Microsoft Access but does not list SQL keywords that are not supported. This is so because there are **over 200 ANSI SQL keywords that Microsoft Access does not support**. Rather than list them all, Microsoft Access documentation lists only the supported keywords.

Das hat sich mit den ADO-Erweiterungen von Jet 4 stark geändert.
Aber manche wichtige Anweisung bzw. Kombinationen wie z.B. COUNT DISTINCT funktionieren noch immer nicht.

Allerdings gibt es kein DBMS, das alle Schlüsselworte unterstützen würde.

Besonderheiten bei der Syntax in Access

- [] nötig bei Worten, die nicht-alphanummerische Zeichen enthalten
- bei Texten doppelte Anführungszeichen statt einfacher wie in ANSI: "blabla" statt 'blabla'
- Platzhalterzeichen * und ? statt ANSI % und _
- Stringverkettung mit & statt ANSI ||

Was bietet Access an Erweiterungen/Besonderheiten?

Einsatzorte für SQL

- QBE d.h. SQL-Ansicht des Abfrage-Editors
- Datenherkunft (RecordSource) von Formularen/Berichten
- Datensatzherkunft (RowSource) von Kombinations-/Listenfeldern
- VBA, Execute von DAO/ADO oder RunSQL von Access-Docmd

VBA

Access/VBA besitzt den sog. **Expression Service** (ES). Er bietet jene Schnittstelle, die die Verwendung vieler VBA-Funktionen in Abfragen ermöglicht.

Anmerkung: Bei Access 2000 wurden die neuen Stringfunktionen (Replace, InstrRev usw) nicht sofort für den Expression Service implementiert. Dadurch funktionieren sie oft nicht in Abfragen. MS hat nicht veröffentlicht, wann das repariert wurde, aber bei manchen Installationen von A00 funktionieren sie. Bei Access 2002 dann ebenfalls.

Umgekehrt können SQL-Statements auf mehrfache Weise in VBA gebastelt und ausgeführt werden (DAO, ADO, DoCmd-Objekt). Das ersetzt z.T. die fehlende Stapelverarbeitung von SQL-Statements bei JET und bietet darüber hinaus weit mehr Programmiermöglichkeiten als die umfangreicheren SQL-Skriptsprache anderer Systeme.

Erweiterungen des SQL-Standards

TRANSFORM, TOP, DISTINCTROW etc. und viele kleinere Abweichungen und Erweiterungen bei ANSI-Standard-Sprachelementen - werden in der Folge besprochen

Auswahlabfragen

Besonderheiten bei JOINS

- in der WHERE-Klausel (seit ewig, SQL-86)

SELECT Foo, Bar
FROM Tabelle1, Tabelle2
WHERE Tabelle1.ID = Tabelle2.ID

- mit JOIN-Schlüsselwort (SQL-92)

SELECT Foo, Bar
FROM Tabelle1 {INNER | LEFT [OUTER] | RIGHT [OUTER]} JOIN Tabelle2
ON Tabelle1.ID = Tabelle2.ID

Schlüsswort OUTER ist optional

JOIN ist besser, weil
- schneller
- SQL-86-Syntax nicht aktualisierbar
- SQL-92 kompatibel

Bei **mehr als 2 gejointen Tabellen** ist die Reihenfolge der Joins egal, weil der JET-Abfrage-Prozessor eigenständig optimiert.

ANSI unterstützt jede **Mischung von INNER und OUTER JOIN** (aufgrund von Klammersetzung), Access nicht, weil der JET-Abfrage-Prozessor beim Verarbeiten die Klammern ignoriert. Regeln für "verbotene" Joins:

- die beim Outer-Join nicht komplett gelieferte (angeschossene) Tabelle kann nicht bei einem Inner-Join dabei sein
- die beim Outer-Join nicht komplett gelieferte (angeschossene) Tabelle kann nicht die angeschossene Tabelle eines anderen Outer-Joins sein

Non-Equi-Joins

Als Verknüpfungsoperator wird nicht wie beim Equi-Join das = verwendet sondern <, > etc. oder BETWEEN...AND
Nur in SQL-Ansicht möglich, nicht in der Entwurfsansicht.

DISTINCTROW

Access-spezifisch
vor A97 immer verwendet bei QBE, ab A97 nicht mehr **à** näher am SQL-Standard

DISTINCT (entspricht der Eigenschaft "Keine Duplikate")
fasst auf Basis einzelner Spalten zusammen **à** nicht aktualisierbar

DISTINCTROW (entspricht der Eigenschaft "Eindeutige Datensätze")
fasst auf Basis ganzer DS zusammen **à** aktualisierbar

(3. Variante wäre ALL = Standard, wenn die beiden nicht verwendet werden)

KB-Artikel zu DISTINCTROW:
ACC97: Changes with DISTINCTROW in Microsoft Access 97
http://support.microsoft.com/default.aspx?scid=kb;en-us;Q168438

ACC2000: "Could Not Delete from Specified Tables" Error Message in Delete Query
http://support.microsoft.com/default.aspx?scid=kb;en-us;Q240098

Tipp: bei nicht aktualisierbaren Abfragen immer mal probieren

TOP und TOP PERCENT

Access-spezifisch
bei mehreren DS im "letzten" Rang werden alle ausgegeben

WITH OWNERACCESS OPTION

Access-spezifisch
Benutzer erbt die Rechte des Besitzer der Abfrage d.h. darf alles, was der Besitzer darf, auch wenn er selber keine Rechte hat. **à** spielt v.a. eine Rolle beim Absichern von BE-Tabellen

entspricht Eigenschaft "Ausführungsberechtigungen" auf "Besitzer"
funktioniert nur mit gespeicherten Abfragen, ansonsten wirkungslos

TRANSFORM (Kreuztabellenabfrage)

Access-spezifisch)

TRANSFORM Aggregatfunktion
SELECT-Statement
PIVOT Spaltenüberschriften-Feld [IN (Wert1, Wert2...Wert-n)]

IN-Klausel dient hier zum Einstellen von fixierten Spalten (auch in den Abfrage-Eigenschaften einstellbar)

Besonderheiten bei UNION

Operator entspricht ANSI
Besonderheit bei Access: Felder müssen nicht den gleichen Typ haben (z.B. Text+Zahl)

Tipp: Option ALL
standardmäßig werden doppelte Einträge eliminiert (quasi ein DISTINCT bzw. GROUP BY)
durch Angabe von ALL werden alle angezeigt und die Performance steigt, weil nix verglichen werden muss

Tipp: Sortierung
mit Feldnamen aus der 1. beteiligten Tabelle oder mit Spaltennummer

Tipp: Option TABLE

statt

```
SELECT * FROM Kunden
UNION
SELECT * FROM Lieferanten
```

```
TABLE Kunden
UNION
TABLE Lieferanten
```

Neu ab A00:
Memo- und Hyperlink-Felder können in einer Union verwendet werden - nur OLE-Ojektfelder weiterhin nicht

Zugriff auf externe Daten

3 Varianten:

- eingebundene Tabellen
- IN-Klausel
- direkter Bezug

eingebundene Tabellen

am einfachsten und performantesten

IN-Klausel

Access:

```
SELECT *
FROM tblMitarbeiter
IN "c:\Daten\Personal.mdb";
```

ISAM:

```
SELECT *
FROM Mitarbeiter
IN "c:\Daten\Personal.xls" "Excel 8.0;";
```

ODBC

```
SELECT *
FROM tblMitarbeiter
IN "" [ODBC;DSN=MyPubs;UID=donkarl;PWD=geheimst];
```

Direkter Bezug

```
SELECT *
FROM [c:\Daten\Personal.mdb].tblMitarbeiter
```

Vorteil: man kann mehrere unterschiedliche Datenquellen anzapfen

```
SELECT Gehalt, Vorname, Nachname
FROM [Paradox 7.X;DATABASE=c:\PdxDaten\Personal].Loehne
INNER JOIN [dBase IV;DATABASE=c:\dBDaten\Personal].Mitarbeiter
ON Loehne.PersNr = Mitarbeiter.PerNr
```

Nachteil: schlechte Performance

Unterabfragen

(auch "Subselects")

Ein SELECT-Ausdruck, in eine der Klauseln der SELECT-Anweisung eingebettet.

Unterabfragen in Spalten-Ausdrücken

z.B. bekommt man die Kunden mit der Anzahl ihrer Bestellungen mit

```
SELECT Kunden.Firma,
(SELECT COUNT(*) FROM Bestellungen WHERE Bestellungen.[Kunden-Code] =
Kunden.[Kunden-Code])
FROM Kunden;
```

"korrelierte" Unterabfragen inkludieren einen Bezug auf die Hauptabfrage

Unterabfragen in der WHERE-Klausel

häufigstes Einsatzgebiet, dort sind als Prädikate möglich:
[NOT] IN
ANY, SOME, ALL
[NOT] EXISTS

Unterabfragen in der FROM-Klausel

Unterabfragen können auch beliebig für Joins mit Tabellen verwendet werden.

Für Unterabfragen als alleinige Datenherkunft in der FROM-Klausel verwendet Access eine schräge Syntax. z.B.

```
SELECT Count(*) AS DistinctCount
FROM (SELECT DISTINCT Bestimmungsland FROM Bestellungen) AS a;
```

im Abfragefenster eingegeben und gespeichert, wird umgewandelt zu

```
SELECT Count(*) AS DistinctCount
FROM [SELECT DISTINCT Bestimmungsland FROM Bestellungen]. AS a;
```

In A97 muss man von vornherein die eckigen Klammern mit Punkt verwenden.

Zur Performance
Ob Unterabfrage oder JOIN schneller ist, lässt sich oft nicht sicher sagen **à** im Einzelfall testen

Data Manipulation Language (DML)

(Aktionsabfragen)

Besonderheiten bei Access

UPDATE (Aktualisierungsabfrage)

```
UPDATE tblKunden
SET tblKunden.Firma = "Neuer Name"
WHERE tblKunden.Firma = "Alter Name";
```

ANSI erlaubt Unterabfragen in der SET-Klausel, Access nicht
dafür erlaubt Access Joins d.h. mehrere Tabellen im UPDATE- (und DELETE-) Statement

SELECT INTO (Tabellenerstellungsabfrage)

nicht ANSI-konform eingesetzt - dort Spaltenwerte in Variablen schreiben

DELETE (Löschabfrage)

nicht ANSI-konforme Schreibweise der DELETE-Klausel mit Tabellen oder Spaltennamen

```
DELETE tblPersonal.*
FROM tblPersonal
WHERE PersonalID = 4711
```

```
DELETE Nachname
FROM tblPersonal
WHERE PersonalID = 4711
```

à trotzdem wird der ganze Datensatz gelöscht

bei ANSI nur:

```
DELETE
FROM tblPersonal
WHERE PersonalID = 4711
```

IN Access möglich, aber nicht ANSI-konform sind JOINS in der FROM-Klausel.

INSERT INTO (Anfügeabfrage)

nicht ANSI-konforme Verwendung von SELECT beim Anfügen einzelner DS:

```
INSERT INTO Kunden (Firma)
SELECT "FooBar";
```

```
INSERT INTO Kunden (Firma)
VALUES "FooBar";
```

Häufige Probleme bei Aktionsabfragen

"**Operation muß eine aktualisierbare Abfrage verwenden**. (Fehler 3073)"
"**Löschen aus angegebenen Tabellen nicht möglich**" Fehler 3086

- DISTINCTROW testen, weil Access sonst manchmal eindeutige Datensätze nicht kapiert
- Falls irgendwo in der Abfrage oder in einer vorgeschalteten Abfrage eine Aggregatfunktion vorkommt (Sum, Count etc.), hält Jet auch die Aktionsabfrage für nicht aktualisierbar (entgegen ANSI) **à** alter Bug.
  Lösungen: Tabellenerstellungsabfrage zwischenschalten und diese temporäre Tabelle verwenden, oder Aggregatfunktion durch Domänen-Aggregat-Funktion ersetzen (DSum, DCount etc.)
- Falls Joins in der Abfrage verwendet wurden, kommt es vor, dass die Abfrage als Select funktioniert, aber nicht als Aktionsabfrage. Einfache Varianten lassen sich mit DISTINCTROW lösen. Manchmal ist der Grund aber ein Mehrfachjoin, bei dem Access das Löschziel nicht erkennt. Dann muss der Join in eine IN-Klausel oder dergl. geändert werden.

UNION-Abfragen können nicht in Aktualisierungsabfragen verwendet werden. (Meldung: Syntaxfehler)
Lösung: Union-Abfrage speichern und in der Aktualisierungsabfrage diese gespeicherte Abfrage verwenden.

DATA DEFINITION LANGUAGE (DDL)

erlaubte DDL-Statements

- CREATE TABLE
- ALTER TABLE
- CREATE INDEX
- DROP

Eingabe nicht nur per Code (DAO, ADO) möglich sondern auch in SQL-Ansicht.

Datentypen

Auszug aus der OH von A00 unter "Äquivalente ANSI SQL-Datentypen"

ANSI SQL-Datentyp	Microsoft Jet-SQL-Datentyp	Synonym	Microsoft SQL Server-Datentyp
BIT, BIT VARYING	BINARY	VARBINARY, BINARY VARYING BIT VARYING	BINARY, VARBINARY
Nicht unterstützt	BIT	BOOLEAN, LOGICAL, LOGICAL1, YESNO	BIT
Nicht unterstützt	TINYINT	INTEGER1, BYTE	TINYINT
Nicht unterstützt	COUNTER	AUTOINCREMENT	
Nicht unterstützt	MONEY	CURRENCY	MONEY
DATE, TIME, TIMESTAMP	DATETIME	DATE, TIME	DATETIME
Nicht unterstützt	UNIQUEIDENTIFIER	GUID	UNIQUEIDENTIFIER

DECIMAL	DECIMAL	NUMERIC, DEC	DECIMAL
REAL	REAL	SINGLE, FLOAT4, IEEESINGLE	REAL
DOUBLE PRECISION, FLOAT	FLOAT	DOUBLE, FLOAT8, IEEEDOUBLE, NUMBER	FLOAT
SMALLINT	SMALLINT	SHORT, INTEGER2	SMALLINT
INTEGER	INTEGER	LONG, INT, INTEGER4	INTEGER
INTERVAL	Nicht unterstützt		Nicht unterstützt
Nicht unterstützt	IMAGE	LONGBINARY, GENERAL, OLEOBJECT	IMAGE
Nicht unterstützt	TEXT	LONGTEXT, LONGCHAR, MEMO, NOTE, NTEXT	TEXT
CHARACTER, CHARACTER VARYING, NATIONAL CHARACTER, NATIONAL CHARACTER VARYING	CHAR	TEXT(n), ALPHANUMERIC, CHARACTER, STRING, VARCHAR, CHARACTER VARYING, NCHAR, NATIONAL CHARACTER, NATIONAL CHAR, NATIONAL CHARACTER VARYING, NATIONAL CHAR VARYING	CHAR, VARCHAR, NCHAR, NVARCHAR

einige Abweichungen zu ANSI z.B. INTEGER erzeugt Long in Access
Viele der Aliase können nur mit dem OLEDB-Provider in ADO verwendet werden.

CONSTRAINT

über die Access-Oberfläche - entgegen ANSI

- keine CHECK-Constraints
- kein NOT NULL

über ADO-JET4-Erweiterungen beides möglich

In der Access-UI gibt es keine Möglichkeit, Referentielle Integrität, Aktualisierungsweitergabe und Löschweitergabe einzustellen. In den ADO-JET 4-Erweiterungen schon.

SQL 92 Erweiterungen in Jet 4

stärkere Anpassung an den ANSI 92 - Standard
die meisten Erweiterungen sind nur per OLEDB-Provider **à** ADO nutzbar, nicht über die Access-Oberfläche oder DAO

Weiterer Grund für die Erweiterungen ist die höhere Kompatibilität beim späteren Umstieg oder Upsizing auf SQL-Server-Backend.

Sortierung auf Memo- und Hyperlink-Felder möglich

Ausführen der Erweiterungen per ADO durch Verwendung der Execute-Methode des Connection-Objektes. z.B.

```
Dim cnn As ADODB.Connection
Dim strsql As String

Set cnn = Application.CurrentProject.Connection

strsql = "CREATE TABLE tblKunden (" & _
         "Firma VARCHAR, " & _
         ' weitere Felder etc.

cnn.Execute strsql
cnn.Close
Set cnn = Nothing
```

CREATE TABLE und ALTER TABLE

erweitert um

- Spaltenänderungen
- Standardwerte
- Check Constraints
- Referentielle Integrität
- schnelle Fremdschlüssel
- Unicode-Komprimierung
- Startwert und Schrittweite bei Autowert

Spaltenänderungen
Datentyp und Feldgröße eines Feldes per Code ändern z.B.
ALTER TABLE MeineTabelle ALTER COLUMN MeinFeld VARCHAR(100)

Standardwerte
z.B.
CREATE TABLE tblArtikel (Lagerbestand Long DEFAULT 100);

Check Constraints
- ähnlich wie Gültigkeitsregel im Access-Tabellen-Entwurfsfenster
- mächtiger, weil mehrere Tabellen involviert sein können (ähnlich wie Trigger)
- aber keine benutzerdefinierte Meldung (wie bei Gültigkeitsmeldung)

z.B.
```
ALTER TABLE tblRechnungen
ADD CONSTRAINT CheckBetrag
CHECK (Betrag >= 0)
```

Referentielle Integrität

z.B.
```
ALTER TABLE tblBestellungen
 ADD CONSTRAINT FK_tblBestellungen
 FOREIGN KEY (KundenID) REFERENCES
 tblKunden (KundenID)
 ON UPDATE CASCADE
 ON DELETE CASCADE
```

Fügt Aktualisierungs- und Löschweitergabe beim Fremdschlüssel KundenID hinzu.

Startwert und Schrittweite bei Autowert
z.B.
```
CREATE TABLE tblUICounterDataTypes (
   Field1 AUTOINCREMENT(10,5),
```

Statt AUTOINCREMENT sind auch IDENTITY oder COUNTER möglich

Views und Stored Procedures

keine neuen Jet-Objekte sondern angepasste Jet-Abfragen
in Access 2000 nicht sichtbar im Datenbankfenster (erst in Access 2002)

Views können alle Select-Varianten einer normalen Abfrage enthalten

```
CREATE VIEW MeineNeueView AS
SELECT Vorname, Nachname
FROM tblPersonal
```

keine ORDER BY-Klausel bei Views

Anwendung wie Access-Abfragen

Löschen mit
```
DROP VIEW MeineNeueView
```

Stored Procedures können SELECTs, DML- und DDL-Statements enthalten
(Auswahl-, Aktions-, Datendefinitons-Abfragen)

```
CREATE PRODECURE SPName [Parameter1, Parameter2] AS
SELECT....oder INSERT INTO etc.
```

```
CREATE PRODECURE MeineNeueSP (lngPersID) AS
SELECT *
FROM tblPersonal
WHERE PersID = lngPersID
```

ausführbar mit dem EXECUTE-Statement (wie z.B. SQL Server SPs)

EXECUTE MeineNeueSP 17

Löschen mit
DROP PROCEDURE SPName

Transaktionen

bisher nur per DAO und ADO

BEGINN TRANSACTION

COMMIT [TRANSACTION]

ROLLBACK [TRANSACTION]

Sicherheit

bisher nur per DAO und ADOX

neuen Benutzer anlegen
CREATE USER MeinNeuerUser SeinPasswort SeinePID

neue Benutzergruppe anlegen
CREATE GROUP MeineNeueGruppe IhrePID

Benutzer eine Gruppe hinzufügen
ADD USER MeinUser1, MeinUser2, MeinUser-n TO MeineGruppe

Passwort ändern
ALTER USER MeinUser PASSWORD NeuesPasswort AltesPasswort

Datenbankkennwort anlegen/ändern
ALTER DATABASE PASSWORD NeuesPasswort AltesPasswort

z.B. wenn es vorher keines gab
ALTER DATABASE PASSWORD ursupergeheim Null

Benutzer löschen
DROP USER Benutzer1 [, Benutzer2, Benutzer-n]

Gruppe löschen
DROP GROUP Gruppe1 [, Gruppe2, Gruppe-n]

Rechte vergeben und entziehen

GRANT Berechtigung1 [, Berechtigung2, Berechtigung-n] ON
TABLE Tabellenname | OBJECT Objektname | CONTAINER Containername
TO Konto1 [, Konto2, Konto-n]

d.h. entweder einzelne Tabelle, einzelnes Ojekt oder einen der Container:

Table (enthält: Tabellen, Abfragen, Views, Stored Procedures)
Forms

Reports
Modules
Scripts

Berechtigungen

SELECT
INSERT
UPDATE
DELETE
DROP
CREATE
SELECTSECURITY
UPDATESECURITY
DBPASSWORD
UPDATEIDENTITY
SELECTSCHEMA
SCHEMA
UPDATEOWNER

z.B.
GRANT SELECT, UPDATE
ON TABLE tblPersonal
TO Users

REVOKE Berechtigung1 [, Berechtigung2, Berechtigung-n] ON
TABLE Tabellenname | OBJECT Objektname | CONTAINER Containername
TO Konto1 [, Konto2, Konto-n]

Performance

Showplan

Textdatei, die den Ausführungsplan der Abfrage zeigt. Showplan wird von MS nicht offiziell unterstützt. Es zeigt v.a. die Verwendung von Indizes und ob Rushmore-Technik verwendet werden konnte oder ob es zu Tablescans und dergl. kam.

Aktivierung für Jet 3.5 und 4.0:

In der Registry neuen Schlüssel hinzufügen:
`\\HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\JET\3.5\Engines\Debug`
bzw.
`\\HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\JET\4.0\Engines\Debug`

neue Zeichenfolge unter diesem Schlüssel hinzufügen:
JETSHOWPLAN

(Großschreibung beachten)

Werte für die Zeichenfolge:
ON = schaltet Ausgabe in Datei ein
OFF = schaltet Ausgabe in Datei aus

Bei ON wird im Standardverzeichnis von Access eine Textdatei SHOWPLAN.OUT erzeugt. Nach jeder Neukompilierung (Ändern der Abfrage oder Komprimieren der DB) wird bei Erstausführung der Abfrage der neue Ausführungsplan in der Shoplan-Datei hinzugefügt.

Ein ähnliches, aber komplizierter zu nutzendes Feature ist die **ISAMStats Function**, die Auskunft über Schreib-, Lese-, Sperr- und Caching-Prozesse von JET gibt. Programmierbeispiel zu finden im "Microsoft Jet 3.5 Performance White Paper" (s. Quellen/Webseiten).

Rushmore

Seit Access 2.0 eingebaute (in Jet 3.5 verbesserte) Optimierungstechnik zur besseren Ausnutzung von Indizes bei Kriterien. v.a. Index-Intersection, Index Union und COUNT(*)

s. Quellen/OH und Rushmore Whitepaper unter Quellen/Webseiten
s. OH "Einfache und komplexe optimierbare Ausdrücke und Rushmore-Optimierung"

Vor Rushmore (Access 1.x) konnten nur einfache Ausdrücke optimiert werden d.h. Index eines Feldes zum Suchen von Datensätzen verwendet werden.

Einfache optimierbare Ausdrücke
(s. OH "Einfache optimierbare Ausdrücke - Beispiele")

z.B. Feld Nachname ist indiziert, nur indiziertes Feld (oder 1. Feld in einem zusammengesetzten Index) wird als Kriterium verwendet

Nachname = "Huber"
Nachname IN ("Schwanke", "Donaubauer", "Becker")

Mit Rushmore können mehrere Indizes kombiniert werden.

Komplexe optimierbare Ausdrücke

Vorraussetzungen
- Der Ausdruck verwendet den Und-Operator oder den Oder-Operator.
- Beide Kriterien bestehen aus einfachen optimierbaren Ausdrücken.
- Beide Ausdrücke enthalten indizierte Felder. Die Felder können einzeln indiziert oder Teil eines zusammengesetzten Index sein.

legendäre Rushmore-Beispiele für Index Union und Index Intersection von MS:
Bevölkerungsstatistik mit indizierten Feldern für Körpermerkmale

hair_color = “blonde” AND eye_color = “blue”

Die beiden Indizes werden hintereinander verwendet und die Suchergebnisse kombiniert d.h. die Schnittmenge ermittelt.

hair_color = “black” OR eye_color = “green”

Die beiden Indizes werden hintereinander verwendet und die Suchergebnisse kombiniert d.h. die Vereinigungsmenge ermittelt.

JET–interne Vorgänge und die Query Engine

Verarbeitung einer Abfrage ist eine Serie von komplexen Analyse- und Optimierungsschritten

Parsing Phase
Lesen und Interpretieren des SQL-Statements

Optimization Phase
Erstellen eines möglichst effizienten Ausführungsplans für das SQL-Statement

Die Jet Query Engine besteht aus folgenden Teilen:

Parser/Binder
- zerlegen
- binden an Tabellenfelder
- Syntax prüfen
- umwandeln in interne Abfrageobjektdefinition

Folder
- schiebt Ausdrücke den Query Tree hoch
- top of the query tree = root **à** keine Ausdrücke mehr auf verworfene DS

Splitter
- spaltet zusammengesetzte Abfragen in Einzelteile auf
- schiebt sie den Query Tree hoch
- schmeißt die zusammengesetzten Abfrageobjekte

Flattener
- schnappt Splitter-Ergebnisse und kombiniert Joins, Order By-, Where-Klauseln

Optimizer
- kostenbasiert = je länger, je teurer
- verwendet Statistiken, um den effizientesten Ausführungsplan zu finden
  oft ungenau, veraltet, bei ODBC problematisch, beim Komprimieren aktualisiert v.a. Daten in aufeinanderfolgenden Speicherseiten

Post-Processor
- bringt kompilierte Abfrage in neue, saubere, kleinere Struktur

Join Processor
- führt die kompilierte Abfrage aus
- verschiedene, komplexe Join-Strategien

Performance-Tipps

s. Quellen/OH und "Microsoft Jet 3.5 Performance White Paper" von Kevin Collins (Quellen/Webseiten)

Komprimieren und Neuausführen
Komprimieren der DB bringt Neuanordnung der DS in benachbarten Speicherseiten und Neusortierung nach dem Primärschlüssel oder eindeutigen Index sowie Neuaufbau der Indexseiten. Es wird außerdem ein Flag zur Neukompilierung der Abfragen gesetzt.
Ausführen der Abfrage nach dem Komprimieren bringt dann die Neukompilierung der Abfrage. Ein neuer Ausführungsplan aufgrund der aktuellen Datenstatistiken wird erstellt.

Indizieren
Möglichst alle Kriterien-, Join- und Sortier-Felder indizieren. Testen, ob Rushmore verwendet werden kann (Showplan).
Achtung vor doppelten Indizierungen bei Fremdschlüsselfeldern (werden von Jet automatisch beim Erstellen von Beziehungen indiziert).

Richtig indizieren
Zuviel oder schlecht indizieren kann bes. bei Aktionsabfragen Zeit kosten und große Anzahl von gesperrten Datensätzen verursachen.
Daumenregeln:
- keine Felder mit hoher Duplizität indizieren (Ja/Nein-Felder, Land etc.)
- Nicht bloß zur Ausnutzung von Rushmore indizieren bzw. Ergebnis testen. Performance ist optimal, wenn aus den kombinierten Index-Ergebnissen möglichst eindeutige Datensätze resultieren

Kleinstmögliche Feldtypen verwenden
besonders bei Feldern, über die gejoint wird

Nur benötigte Felder anzeigen
z.B. Felder, die nur für Kriterien da sind, auf "nicht anzeigen" stellen

Abfragen speichern
statt SQL-Strings einzusetzen
gespeicherte Abfrage muss nicht ständig neu evaluiert werden

berechnete Felder vermeiden
besonders in Unterabfragen und WHERE-Klauseln
erschweren das Optimieren für JET
berechnete Felder wenn möglich in Formulare oder Berichte verlagern

wenige Gruppierungen und aus Tabelle mit Berechnungen
statt Gruppierung ist Erster Wert oft schneller, bei Berechnungen sollte die Gruppierung möglichst auf Feld der gleichen Tabelle gemacht werden

Kriterien bei berechneten und nicht indizierten Feldern vermeiden
außerdem versuchen, Rushmore zu ermöglichen

Domänen-Aggregat-Funktionen vermeiden
können von JET nicht optimiert werden, möglichst durch Joins oder Unterabfragen ersetzen

feste Spaltenüberschriften bei Kreuztabellen einsetzen

Sortierung nur, wenn unbedingt notwendig
in Access sind bis zu 10 ORDER BY möglich (Horror! ;-)

Sortierung nicht auf Ausdrücken
sondern auf Spalten (am besten indiziert)

z.B. nicht
ORDER BY Vorname & " " & Nachname
sondern
ORDER BY Vorname, Nachname

DAO-Objekte explizit schließen
Szenario:
DAO-Objekte werden geöffnet und danach im Code eine gespeicherte Abfrage ausgeführt. DAO-Objekte werden nicht explizit geschlossen. Beim erneuten Ausführen der Abfrage wird sie evtl. neu kompiliert (kostet Zeit und bläht die MDB auf).
Grund:
Gespeicherte Abfragen referenzieren best. Speicherstellen zwecks Verwendung des VBA Expression Services. Wenn dieser Speicher beim nächsten Ausführen noch blockiert ist, wird manchmal rekompiliert.

Performance von gespeicherter Parameter-Abfrage mit temporärer Abfrage vergleichen
Bei Parametern (bes. auf indizierte Felder), weiß Jet den Parameterwert nicht vorher. Also wird Ausführungsplan geschätzt und geht öfter daneben. Ausführen von SQL-Statement per Code kann daher schneller sein.

Quellen

Bücher

SQL-Kapitel der ADHs z.B.
Access 2002 Developer's Handbook Set
Ken Getz, Paul Litwin, Mike Gunderloy
2002, Sybex, ISBN 0782140114

Microsoft Jet Database Engine Programmer's Guide
Kevin Collins, Microsoft Press 1998
v.a. Kapitel 3 und 4, nur mehr schwer erhältlich - am ehesten in alten MSDN, Technet

Running Microsoft Access 2000
John Viescas
MSPress 1999, ISBN 1-57231-934-8
Kapitel 11 - "Advanced Query Design—SQL"

einführende/Systemvergleichende Bücher:

GoTo SQL
Michael J. Hernandez, John L. Viescas
Addison-Wesley, 2001, ISBN 3-8273-1772-X
bzw. Original: SQL For Mere Mortals

Nitty Gritty SQL
Heinz-Gerd Raymans
Addison-Wesley, 2002, ISBN 3-8273-1990-0
Vergleich Access, MS SQL-Server, Oracle, MySQL

SQL - der Schlüssel zu relatonalen Datenbanken
Gregor Kuhlmann, Friedrich Müllmerstadt
rororo, 2001, ISBN 3-499-61206-2
Vergleich Access, Oracle, Informix, DB2

Webseiten

ACC97: Microsoft Jet 3.5 Performance White Paper Available in Download Center
http://support.microsoft.com/default.aspx?scid=kb;en-us;Q162701

ACC2: Rushmore Query Optimization Paper Available in Download Center
http://support.microsoft.com/default.aspx?scid=kb;en-us;Q128388

Microsoft Jet SQL for Access 2000
Acey James Bunch, Microsoft Corporation, February 2000
Applies To: Microsoft® Access 2000

Überblick:
http://msdn.microsoft.com/code/default.asp?url=/code/sample.asp?url=/msdn-files/026/002/580/msdncompositedoc.xml

Fundamental Microsoft Jet SQL for Access 2000
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnacc2k/html/acfundsql.asp

Intermediate Microsoft Jet SQL for Access 2000
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnacc2k/html/acintsql.asp

Advanced Microsoft Jet SQL for Access 2000
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnacc2k/html/acadvsql.asp

Access-Online-Hilfe

oft unterschätzt, z.B.

- Register Inhalt
  Microsoft Jet SQL-Sprachreferenz
- Äquivalente ANSI SQL-Datentypen
- Optimieren der Abfrageleistung
- Einfache und komplexe optimierbare Ausdrücke und Rushmore-Optimierung
- Tipps zum Optimieren der Leistung externer SQL-Datenbanken