Tipps & Tricks

Access, MySQL, SQL-Datenbanken

1. Access ab 2010 Performance-steigerung einer Access-Datenbank

Access-Datenbanken werden im Laufe der Zeit immer langsamer. Das liegt am Speicherverhalten von Access. Einmal reservierten Platz gibt das Programm nicht freiwillig wieder her. Auch Programmiercode behindert die Performance, da dieser bei jedem Ausführen neu kompiliert wird. Ein paar Tricks steigern die Performance von Access gewaltig.

Zunächst sorgen Sie dafür, dass Ihre Datenbank beim Schließen komprimiert wird. Dadurch wird nicht benötigter Speicherplatz wieder freigegeben. Sie finden den Befehl dafür im Register *Datei*, *Optionen*. Wählen Sie dort die Kategorie *Aktuelle Datenbank* aus und setzen Sie einen Haken in das Kontrollkästchen *Beim Schließen komprimieren*.

Nach diesem Leistungsschub gehen Sie noch einen Schritt weiter und kompilieren Ihre Datenbank als eine ausführbare Datei. Dadurch werden alle Module mit VBA-Code kompiliert und der komplette bearbeitbare Quellcode entfernt. Zusätzlich wird die Zieldatenbank komprimiert. Der VBA-Code behält seine Funktionalität, und die Datenbank funktioniert weiterhin wie gewohnt, nur mit erhöhter Leistung. Zusätzlich zum nicht mehr vorhandenen VBA-Code werden Formulare und Berichte geschützt. Diese können weder verändert noch kopiert werden. Der Anwender kann dann auch keine neuen Formulare oder Berichte erstellen.

Sie als Entwickler müssen allerdings die Originaldatenbank behalten, damit Sie weiterhin Zugriff auf alle Objekte haben. Der Anwender hat dann aber eine kleinere und schnellere Datenbank.

Durch das Kompilieren Ihrer Access-Datenbank verkleinern Sie deren Dateigröße und erhöhen die Performance. Außerdem schützen Sie Formulare und Berichte vor Änderungen.

Zum Erstellen einer kompilierten Datenbank wählen Sie im Register *Datei* den Befehl *Speichern unter* und dann den Befehl *ACCDE erstellen* aus. Sinnvollerweise sollten Sie dieser Datebank dann auch einen von Ihrer Entwicklerversion abweichenden Namen geben.

2. Access ab 2010 Gelöschte Daten sichern

In Access-Datenbanken gibt es keine Rollback-Funktionen. Gelöschte Datensätze sind endgültig verloren. Mit einem Makro haben Sie die Möglichkeit, gelöschte Datensätze direkt in eine Archivdatei zu schreiben. Dieses sollten Sie aus Sicherheitsgründen auch tun.

Ab Access 2010 gibt es in den Tabellen die Datenmakros. Diese Makros reagieren auf Ereignisse, die in Tabellen auftreten und sind somit sehr gut dazu geeignet, gelöschte Daten zu archivieren. Die Datenmakros werden immer gestartet, unabhängig davon, ob das Tabellenereignis in der Datenblattansicht einer Tabelle, einer aktualisierbaren Abfrage, in einem Formular oder durch VBA-Befehle ausgelöst wird. Das soll am Beispiel einer Kundentabelle demonstriert werden.

```
Datensatz erstellen in ArchivFirmen
    FestlegenFeld
        Name ArchivFirmen.Firmenname
        Wert = alt.[Firmenname]
    FestlegenFeld
        Name ArchivFirmen.Anprechpartner
        Wert = alt.[Anprechpartner]
    FestlegenFeld
        Name ArchivFirmen.Ort
        Wert = alt.[Ort]
    FestlegenFeld
        Name ArchivFirmen.Postleitzahl
        Wert = alt.[Postleitzahl]
    FestlegenFeld
        Name ArchivFirmen.Rechnungsadresse
        Wert = alt.[Rechnungsadresse]
    FestlegenFeld
```

Mit den neuen Datenmakros erstellen Sie *StoredProcedures*, die bei dem jeweiligen Ereignis ausgeführt werden.

Erstellen Sie zunächst eine Kopie der Struktur der zu archivierenden Tabelle. Klicken Sie die gewünschte Tabelle dazu mit der rechten Maustaste an und wählen Sie die Befehle *Kopieren* und *Einfügen* aus. Im daraufhin eingeblendeten Dialogfenster benennen Sie die Tabelle um, zum Beispiel in *ArchivFirmen*. Aktivieren Sie die Option *Nur Struktur*.

Öffnen Sie diese Tabelle in der Entwurfsansicht. Entfernen Sie alle vorhandenen Primärschlüssel. Erstellen Sie ein zusätzliches Feld mit dem Namen *LogID* und der Eigenschaft *Autowert*. Fügen Sie ein weiteres

MySQL: Die wichtigsten Tuning-Parameter

Parameter	Auswirkung	Empfehlung
table_cache	Der Table-Cache reserviert Speicher für die häufig angesprochenen Tabellen.	mindestens Anzahl der Tabellen
query_cache_size	In dem Speicherbereich werden die SQL-Statements gespeichert. Dadurch stehen die Daten bei einem erneuten Zugriff sofort zur Verfügung. Bei Updates/Inserts/Deletes wird der gespeicherte Query gelöscht.	abhängig von der Anzahl der Querys (mysql> SHOW STATUS LIKE "qcache%";
key_buffer_size	der Speicher, der für die referenziellen Keys reserviert wird	Anzahl der festgelegten Indizies
sort_buffer_size	der Speicher, der für die Sortierung reserviert wird	
read_buffer_size	der Speicher, der für das Lesen von Festplatte reserviert wird	systemabhängig

Feld mit dem Namen *LogDatum* und dem Felddatentyp *Datum/Uhrzeit* hinzu. Speichern und schließen Sie die Archivtabelle.

Öffnen Sie Ihre Kundentabelle und aktivieren Sie das Register *Tabelle*. In der Gruppe *Nachfolgeereignisse* aktivieren Sie das Symbol *Nach Löschung*. Die Entwurfsansicht der *Makrotools* wird eingeblendet.

Auf der rechten Seite des Makro-Editors wird Ihnen ein Aktionskatalog angezeigt, der alle Befehle enthält, die Sie für das zu erstellende Datenmakro einsetzen können. Klicken Sie im linken Fensterbereich auf den Pfeil im Kombinationsfeld *Neue Aktion hinzufügen* und wählen Sie den Eintrag *DatensatzErstellen* aus. Es wird ein neuer Datenblock eingefügt. Geben Sie im Kombinationsfeld *Datensatz erstellen in* den Tabellennamen *ArchivFirmen* ein. Klicken Sie im Datenblock *Neue Aktion hinzufügen* auf den Eintrag *FestlegenFeld*. Geben Sie im Feld *Name* zum Beispiel den Ausdruck *ArchivFirmen.Firmenname* und im Feld *Wert* den Ausdruck *[Alt].[Firmenname]* ein. Hiermit legen Sie fest, dass der alte Firmenname aus dem bereits gelöschten Datensatz der Kundentabelle in das Feld *Firmenname* der Tabelle ArchivFirmen übertragen wird. Beim Eingeben des Ausdrucks werden Sie durch Intellisense-Funktionen unterstützt. Verfahren Sie für die weiteren zu archivierenden Felder analog.

Zum Schluss soll noch der Zeitpunkt gespeichert werden, wann der Datensatz gelöscht wurde. Wählen Sie dazu im Datenblock *Neue Aktion hinzufügen* nochmals den Eintrag *FestlegenFeld* aus. Tippen Sie in das Feld *Name* den Ausdruck *ArchivFirmen.LogZeit* und im Feld *Wert* den Ausdruck *JETZT()* ein.

Speichern und testen Sie zum Abschluss das Datenmakro. Gelöschte Datensätze werden jetzt in Ihre Archivtabelle geschrieben. Mit diesem Makro schaffen Sie Datensicherheit, da Sie versehentlich gelöschte Datensätze jederzeit wieder in Ihre Kundentabelle zurückspielen können.

3. MySQL ab Version 4 Eine MySQL-Datenbank beschleunigen

Sie beschleunigen eine Datenbank grundsätzlich durch das Erzeugen von Indexen. Aber auch Queries können eine Datenbank ausbremsen. Bei MySQL müssen Sie insbesondere noch auf die Einstellungen achten.

Das Setzen von Indexen ist mit einem Befehl schnell abgeschlossen.

```
CREATE UNIQUE INDEX meinIndex
ON
meineTabelle (
MeinFeld
)
```

Die Tabelle *meineTabelle* erhält den INDEX mit dem Namen meinIndex. Durch die Option *UNIQUE* ist der Index eindeutig. Es dürfen keine Felder mit gleichem Wert in der Spalte MeinFeld vorkommen.

Langsame Queries loggen Sie durch folgende Einträge in der Konfigurationsdatei my.cnf:

```
log-slow-queries
long_query_time = 3
```

Das Logfile erscheint dann im MySQL-Datenverzeichnis mit dem Dateinamen *Servername-slow.log*. Der Parameter *long_query_time* legt fest, ab wie viel Sekunden eine Query als langsam gilt.

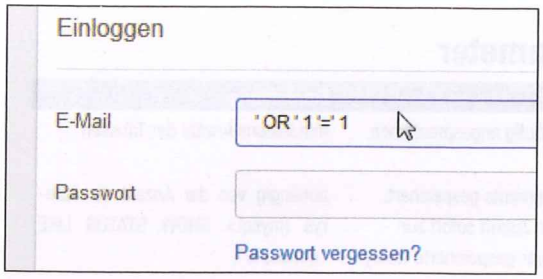
Für eine weitere Beschleunigung Ihrer Datenbank überprüfen Sie die my.cnf-Konfigurationsdatei auf dem Query Cache. Sie sollten diesen Cache akti-

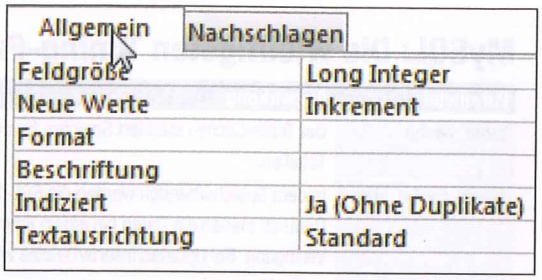
Überprüfen Sie Ihre Abfragen auf Zeitfresser und optimieren Sie überladene Querys.

```
SELECT DISTINCTROW Firma.Firma, Rechnungskopf.Rechnungsnummer, Format$([Rechnungskopf].[Rechnungsdatum],"yyyy") AS Rechnungsjahr, Rechnungskopf.Status,
Sum(Rechnungszeilen.GEuro) AS [Summe von Gesamtpreis], Rechnungszeilen.MWsT, Rechnungszeilen.Leistung, Rechnungskopf.Rechnungsdatum, Rechnungskopf.Bemerkungen
FROM (Rechnungskopf INNER JOIN Firma ON Rechnungskopf.Firma = Firma.[Kunden-Nr]) INNER JOIN Rechnungszeilen ON Rechnungskopf.Rechnungsnummer =
Rechnungszeilen.Rechungsnummer
GROUP BY Firma.Firma, Rechnungskopf.Rechnungsnummer, Format$([Rechnungskopf].[Rechnungsdatum],"yyyy"), Rechnungskopf.Status, Rechnungszeilen.MWsT,
Rechnungszeilen.Leistung, Rechnungskopf.Rechnungsdatum, Rechnungskopf.Bemerkungen,
Year([Rechnungskopf].[Rechnungsdatum])*12+DatePart("m",[Rechnungskopf].[Rechnungsdatum])-1, Year([Rechnungskopf].[Fällig])*12+DatePart("m",[Rechnungskopf].[Fällig])-1
HAVING (((Firma.Firma)="dataport") AND ((Rechnungszeilen.MWsT)=0));
```

Links: Versuch einer Injection mit Sonder- bzw. Steuerzeichen.

Rechts: Durch das Indizieren von Feldern beschleunigen Sie eine Datenbank.





vieren, dadurch erhöhen Sie die Performance Ihrer Datenbank spürbar. Kontrollieren Sie auch den Key_buffer. Stellen Sie diesen so groß ein, dass alle Indizies in diesen Buffer reinpassen. Der Table Chache sollte der Anzahl der Tabellen entsprechen. Falls Sie mit MySQL nur den localhost nutzen, deaktivieren Sie die Netzwerkfunktionalität mit *skip-networking*.

4. SQL-Datenbanken **Sicherheit: Verhindern von MySQL-Injections**

Online-Datenbanken werden immer wieder angegriffen. Besonders beliebt sind MySQL-Injections. Davor sollten Sie Ihre Datenbank schützen.

Verbindungen zu SQL-Datenbanken sollten möglichst strikt erfolgen. User dürfen nur Zugriff auf die Tabellen haben, die benötigt werden. So können kritische Daten geschützt werden. Wenn Daten nur gelesen werden sollen, darf der SQL-Benutzer auch nur read-only-Berechtigungen besitzen. Daher ist ein Blick in die MySQL-Berechtigungen unbedingt notwendig. Um die Ausnutzung von SQL-Injections zu verhindern, sind eine Reihe von zusätzlichen Maßnahmen zu ergreifen.

Eine der wichtigsten Maßnahmen zur Vermeidung von SQL-Injections ist die sorgfältige Überprüfung und Filterung von Eingaben und Parametern durch die Applikation. Als Erstes überprüfen Sie, ob die übergebenen Daten dem erwarteten Datentyp entsprechen. Wird ein numerischer Parameter erwartet, prüfen Sie das mit der Funktion *is_numeric()*.

Die Filterung muss dafür sorgen, dass Sonderzeichen wie das Quote-Zeichen *(,)*, das Semikolon *(;)* und doppelte Bindestriche *(--)* ignoriert werden. Sonst haben Hacker leichtes Spiel, Ihr System zu übernehmen.

Sicherer ist der Einsatz von *Stored Procedures* beziehungsweise *Prepared SQL-Statements*. Diese werden von vielen Datenbank-Managementsystemen (DBMS) angeboten und sind ursprünglich dazu gedacht, häufiger auftretende Abfragen zu optimieren. Der Vorteil dieser parametrisierten Statements ist, dass Parameter nicht mehr direkt in ein SQL-Statement eingebunden werden. Vielmehr werden diese getrennt vom SQL-Statement separat an die Datenbank übergeben. Das Zusammenführen von Statement und Parametern erfolgt durch das DBMS selbst, wobei die oben genannten Sonderzeichen automatisch maskiert werden. Dadurch machen Sie einem Angreifer das Leben sehr schwer.

Um potenziellen Angreifern keine Anhaltspunkte für Angriffe zu liefern, sollte besonderes Augenmerk darauf gelegt werden, dass Applikationen möglichst keine Fehlermeldungen nach außen ausgeben, die Rückschlüsse auf das verwendete System oder auf die Struktur der dahinterliegenden Datenbank zulassen. Sonst ist dank Suchmaschinen eine Schwachstelle schnell gefunden.

In PHP verhindert zum Beispiel die Funktion *mysql_real_escape_string()* die Übergabe von Sonderzeichen an eine MySQL-Datenbank. Die Funktion maskiert die in dem übergebenen String enthaltenen Sonderzeichen und verhindert so SQL-Injections.

Anstatt der folgenden Syntax:

```
$query = "SELECT * FROM users
WHERE username=
'" . $_POST['username'] . "'
AND password=
'" . $_POST['password'] . "'";
```

sollten Sie diese Syntax verwenden:

```
$query = "SELECT * FROM users
WHERE username=
'" . mysql_real_escape_string($_
POST['username']) . "'
AND password=
'" . mysql_real_escape_string($_
POST['password']) . "'";
```

Peter Schnoor/whs

Beschleunigen Sie Ihre Datenbanken durch das Erstellen von Primärschlüsseln und Verknüpfungen.