

Software-Anleitung

Das Excel-Werkzeug "Tabelle" im Praxiseinsatz

Teil 2: Tabellen konsolidieren mit ODBC und SQL

Ab Version 2007 stellt Excel für die Datenanalyse die Werkzeuge "Tabellen" und "strukturierte Verweise" zur Verfügung. Welche Vorteile diese gegenüber herkömmlichen Excel-Listen und Zellverweisen bieten und wie man strukturierte Verweise verwendet, um Tabellendaten zu analysieren, ist im ersten Teil des Beitrags beschrieben. Doch auch beim Einlesen externer Daten per ODBC (open database connectivity) bringen die Werkzeuge Vorteile. Der zweite und abschließende Teil zeigt, wie Daten per ODBC aus externen Quellen importiert, als Tabellen bearbeitet und mit strukturierten Verweisen analysiert werden.

ODBC ist eine standardisierte Datenbankschnittstelle von Microsoft auf Basis der Datenbanksprache SQL. Für den Datenimport in Excel-Tabellenblätter stehen entsprechende Treiber zur Verfügung, die mit der Installation des Office-Pakets eingerichtet werden. Welche ODBC-Treiber genau auf Ihrem Rechner installiert sind, können Sie im "ODBC-Datenquellen-Administrator" erkennen, den Sie in der Systemsteuerung in der Gruppe "Verwaltung" finden.

Standardmäßig bietet Excel Importmöglichkeiten für Daten aus unterschiedlichen Quellen, u.a. aus Access, dem Web und aus Textdateien. Sie finden die entsprechenden Befehle im Register *Daten* im Bereich *Externe Daten abrufen* (Bild 1). Hinter jeder der Optionen verbirgt sich ein Assistent, der den Anwender durch den Prozess der ODBC-Integration führt. Für einfache Abfragen z.B. aus externen Excel-Tabellenblättern reicht der ODBC-Abfrageassistent "Microsoft Query" aus, den Sie im Bereich *Externe Daten abrufen* über die Auswahl *Aus anderen Quellen* aufrufen.

ODBC und Tabellen

Eine ODBC-Verbindung auf externe Daten lieferte in Excel bis Version 2003 eine Liste, die mit der externen Datenquelle verknüpft war. Ab Excel 2007 ist das Ergebnis eine Tabelle – was die Datenanalyse erleichtert. (Eine ausführliche Beschreibung zu Tabellen und strukturierten Verweisen finden Sie im [ersten Teil dieses Beitrags](#).)

Gegenüber herkömmlichen Listen haben Tabellen folgende Vorteile:

- Tabellen haben einen Namen, der sich automatisch an die Dimension der Tabelle anpasst. Werden neue Daten angefügt, erweitert sich der Bereich, auf den sich der Tabellenname bezieht, automatisch.

Autor

**Ignatz Schels**
Seit 1986 selbst. DV-Dozent und Journalist, Leitet u. konzipiert MS-Project-Seminare u.a. für Daimler, Siemens, T-Systems
Kontakt: info@schels.de
Mehr Informationen unter: projektmagazin.de/autoren

ähnliche Artikel

in der Rubrik:
[Microsoft Excel](#)

Service-Links

 Bücher
[Microsoft Excel](#)

- Die Ergebniszeile summiert oder zählt automatisch einzelne Spaltenwerte.
- Mit internen strukturierten Verweisen können Daten spaltenweise berechnet werden. Die Formeln verwenden keine Bezüge (z.B. `=A$1`), sondern Spaltennamen (z.B. `=[@Budget]`) und sind daher intuitiver zu bedienen.
- Externe strukturierte Verweise verwenden Elemente der Tabelle für Analysen, zum Beispiel einzelne Spalten (`[Spaltenname]`), den Datenbereich (`[#Daten]`), die Kopfzeile (`[#Kopfzeilen]`) oder die Ergebniszeile (`[#Ergebnisse]`).

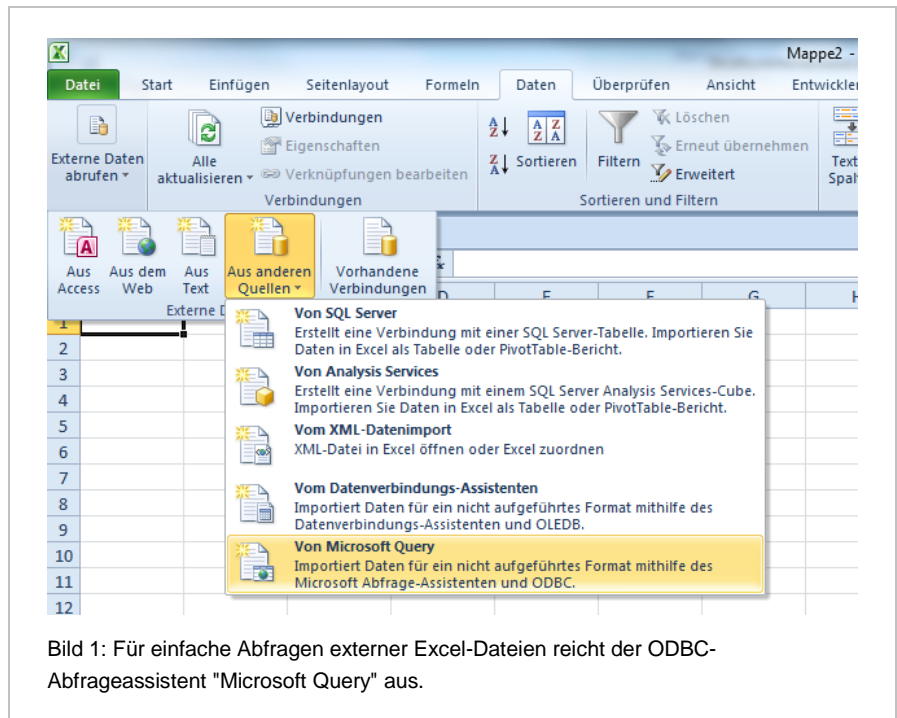


Bild 1: Für einfache Abfragen externer Excel-Dateien reicht der ODBC-Abfrageassistent "Microsoft Query" aus.

Die Vorteile von Tabellen als Ergebnis einer ODBC-Verknüpfung sind:

- Die Tabelle bleibt nicht auf die Spalten aus der Quellenanwendung beschränkt, sondern es können jederzeit weitere Spalten ergänzt werden. Befinden sich in diesen Spalten Formeln, schreibt Excel diese automatisch fort, falls sich durch die Aktualisierung der ODBC-Verbindung der Tabellenbereich vergrößert.
- ODBC-Verknüpfungen sind technisch gesehen SQL-Befehle. Wer diese (einfache) Datenbanksprache beherrscht, kann externe Tabellen gezielt filtern, sortieren oder sogar kombinieren.

Externe Daten per ODBC zusammenführen

Beispiel: Mehrprojektverwaltung

Das Prinzip des Datenimports über ODBC lässt sich am einfachsten an einem Beispiel erläutern (Die zugrunde liegenden Listen finden Sie in der Beispieldatei im Anhang). Im Beispiel sollen Projektlisten aus unterschiedlichen Tabellenblättern einer Excel-Arbeitsmappe zu einer gemeinsamen Liste bzw. Tabelle zusammengeführt werden. Die Listen sind einheitlich aufgebaut und enthalten neben den Spalten *Projekt*, *Beginn* und *Ende* auch Spalten für die *Plankosten* und den jeweiligen *Fertigstellungsgrad* (Bild 2). Da im Beispiel die einzelnen Unternehmensbereiche sowohl die Projektlisten als auch die Fertigstellungsgrade der Projekte ständig updaten, muss die Gesamtübersicht dynamisch auf jede Änderung reagieren und stets den aktuellen Stand darstellen. Außerdem soll in der Gesamtliste aus den Produkten von Plankosten und Fertigstellungsgrad der Earned Value (Fertigstellungswert) für alle Einzelprojekte berechnet werden.

The image shows three overlapping Excel spreadsheets, each representing a project list for a different city: München, Stuttgart, and Frankfurt. Each spreadsheet has five columns: A (Projekt), B (Beginn), C (Ende), D (Plankosten), and E (Fertigstellungsgrad). The spreadsheets are stacked, with München on top, Stuttgart in the middle, and Frankfurt at the bottom.

A	B	C	D	E
1 Projekt	Beginn	Ende	Plankosten	Fertigstellungsgrad
2 Karosseriebau Instandhaltung	01.10.2013	29.01.2014	120.000	80%
3 Verbesserung Elektronik	01.10.2013	03.02.2014	150.000	60%
4 CDK Produktpflege	15.10.2013	22.02.2014	125.000	40%
5 Ausarbeitung Richtlinien Umweltschutz	15.10.2013	22.02.2014	160.000	30%

A	B	C	D	E
1 Projekt	Beginn	Ende	Plankosten	Fertigstellungsgrad
2 Getriebestrang Automatisierung	01.10.2013	31.03.2014	90.500	20%
3 Kapazitätserhebung und Pers.planung	01.11.2013	31.03.2014	135.000	20%
4 Unfallverhütungsvorschriften	15.10.2013	15.03.2014	56.000	20%
5 Entwicklung e-commerce-Lösung	01.12.2013	15.03.2014	260.500	20%

A	B	C	D	E
1 Projekt	Beginn	Ende	Plankosten	Fertigstellungsgrad
2 CDLK Serie II	01.10.2013	30.04.2014	490.000	12%
3 ABM-V Serie VI	15.10.2013	14.05.2014	320.000	20%
4 Kernprozesse optimieren	01.11.2013	01.02.2014	260.000	0%
5 Neubau Lagerhalle Süd	13.11.2013	01.10.2014	1.500.000	0%

Bild 2: Drei Projektlisten in Excel-Tabellenblättern.

Die Projektlisten sind nicht in Tabellen konvertiert und haben auch keine Bereichsnamen. Beides ist zwar möglich, jedoch nicht Voraussetzung für den Datentransfer per ODBC sowie die Analyse mit strukturierten Verweisen.

! Achten Sie darauf, dass die Listen weder überflüssige Zeilen und Spalten enthalten noch Teilergebnisse oder Zwischensummen. Über die Filterfunktionen der ODBC-Verbindung lassen sich zwar z.B. Leerzeilen leicht herausfiltern, am sichersten sind jedoch reine Datenlisten.

Gesamtübersicht anlegen und die erste Liste per ODBC einlesen

- Schließen Sie die Arbeitsmappe mit den drei Projekten, sie darf während des Aufbaus der ODBC-Verbindung nicht aktiv sein.
- Legen Sie für die Analyse der Projekte eine neue Arbeitsmappe an. In der ersten leeren Tabelle starten Sie den ODBC-Transfer mit dem Befehl *Daten / Externe Daten abrufen / Aus anderen Quellen*. Wählen Sie *Von Microsoft Query*.
- In der Liste der ODBC-Datenquellen entscheiden Sie sich für den ODBC-Treiber für Excel. Ab Version 2007 von Excel heißt dieser *Excel Files*, zuvor hatte er die Bezeichnung *Excel Dateien*. Die Option *Query-Assistenten zur Erstellung / Bearbeitung von Abfragen verwenden* muss aktiviert sein, damit der Assistent Sie durch die Prozedur führt.
- Die Verbindung wird hergestellt, wählen Sie die Arbeitsmappe mit den Quelldaten aus. Jetzt wird im Normalfall die Fehlermeldung *Diese Datenquelle enthält keine sichtbaren Tabellen* erscheinen (Bild 3). Würde die Arbeitsmappe Tabellen oder benannte Bereiche enthalten, könnte Microsoft Query diese erkennen und auflisten. "Reine" Tabellenblätter stuft Microsoft Query dagegen als nicht auswertbar ein.
- Bestätigen Sie mit *OK* und schalten Sie im Fenster des Assistenten unter *Optionen* die Systemtabellen ein (Bild 3). Dadurch erhalten Sie auch die Liste der Tabellenblätter zur Auswahl.

- Sie können die erste Projektliste ("Frankfurt") markieren und über die Pfeilsymbole einzelne Spalten oder die gesamte Feldliste nach rechts in Ihre Abfrage holen (Bild 4). Bestätigen Sie mit *Weiter*, wenn die Feldliste komplett ist.
- Die nächsten Schritte des Query-Assistenten bieten Filterung und Sortierung an. Zum Filtern holen Sie einen oder mehrere Spaltentitel in das Filterfeld und bestimmen die Filterkriterien. Um beispielsweise Leerzeilen auszuschließen, wählen Sie diesen Filter:
Zu filternde Spalte: "Projekt";
Filter: "Ist nicht Null"
Bestätigen Sie mit *Weiter* und sortieren Sie im nächsten Schritt die Liste nach dem ersten Feld *Projekt*.
- Im darauffolgenden Schritt wählen Sie die Option *Daten an Microsoft Excel zurückgeben* und klicken auf *Fertig stellen*, um den Datenimport abzuschließen.
- Als letzten Schritt geben Sie im Dialog "Daten importieren" das Zielformat "Tabelle" an sowie den Ort, an dem die Daten eingefügt werden sollen (hier: erste Zelle des bestehenden Tabellenblatts) (Bild 5).

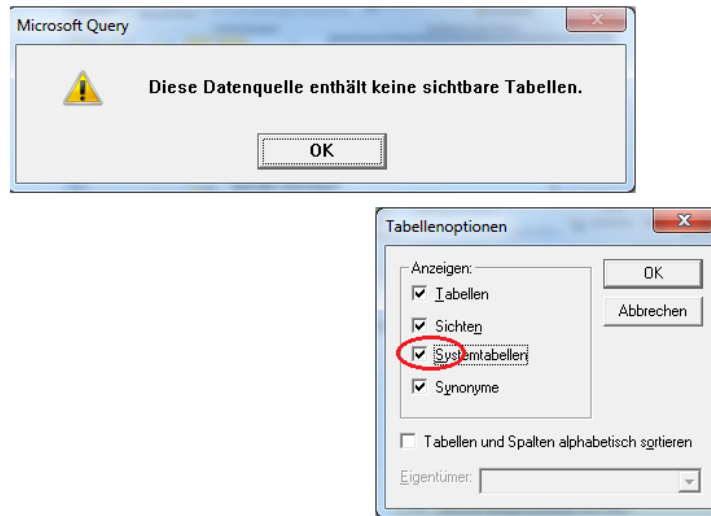


Bild 3: Microsoft Query behandelt Tabellenblätter als "Systemtabellen". Deren Anzeige muss zunächst aktiviert werden.

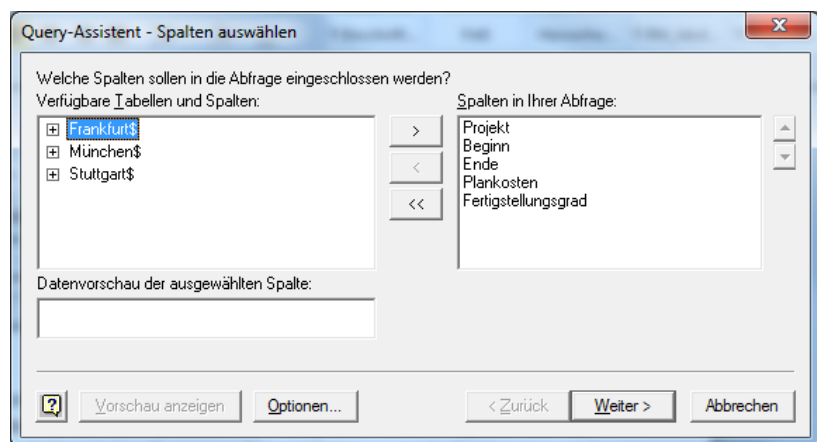


Bild 4: Tabellenblätter und Spaltenauswahl.

Das Ergebnis des ODBC-Datentransfers ist eine Tabelle mit den verknüpften Daten aus der ersten Projektliste (Bild 6). Passen Sie zum Abschluss noch die Zahlenformate der einzelnen Spalten an, indem Sie kürzere Datumsformate für Beginn und Ende, Tausenderpunkte für die Plankosten sowie Prozentzahlen für den Fertigstellungsgrad wählen.

Tabelle formatieren und Verbindung überprüfen

Excel formatiert die Tabelle automatisch mit einer Tabellenformatvorlage und versieht sie mit dem Tabellennamen *Tabelle_Abfrage_von_Excel*. Beides können Sie über das Register *Tabellentools / Entwurf* ändern. Die Tabellenformatvorlagen finden Sie in der gleichnamigen Gruppe rechts außen im Menüband, den Namen der Tabelle passen Sie unter *Eigenschaften / Tabellenname* an. Ändern Sie den vorgegebenen Namen in: *tbl_Projekte_Frankfurt*

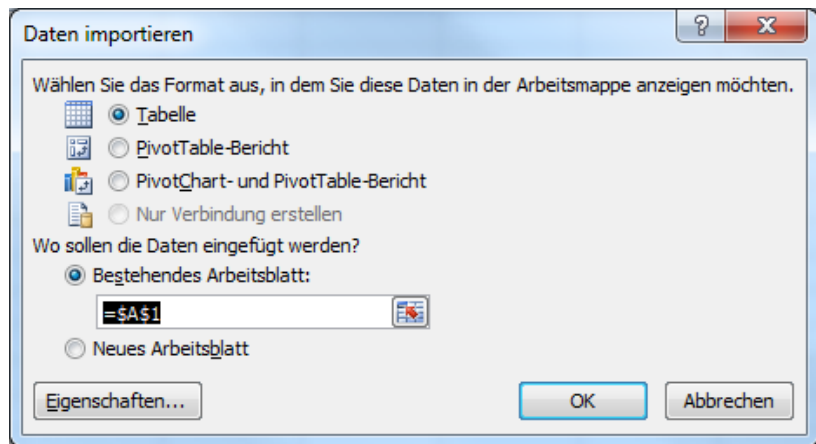


Bild 5: Die Daten werden mit dem Einfügen in eine Tabelle konvertiert.

Projekt	Beginn	Ende	Plankosten	Fertigstellungsgrad
ABM-V Serie VI	15.10.13	14.05.14	320.000	20%
CDLK Serie II	01.10.13	30.04.14	490.000	12%
Kernprozesse optimieren	01.11.13	01.02.14	260.000	0%
Neubau Lagerhalle Süd	13.11.13	01.10.14	1.500.000	0%

Bild 6: Die neue Tabelle ist per ODBC mit dem Original verknüpft.

Die Verbindung ist halbdynamisch, d.h. Sie müssen das Ergebnis aktualisieren, wenn sich die Quelldaten geändert haben:

- Setzen Sie den Zellzeiger in die Tabelle.
- Wählen Sie entweder *Tabellentools / Entwurf / Externe Tabellendaten / Aktualisieren* oder alternativ *Daten / Verbindungen / Alle aktualisieren*.

In den Verbindungseigenschaften unter *Tabellentools / Entwurf / Eigenschaften* legen Sie fest, ob nach der Aktualisierung der Daten die Sortierung, das Format und die Spaltenbreiten erhalten bleiben. Außerdem definieren Sie, was passiert, wenn sich die Anzahl der Zeilen im Datenbereich ändert (Bild 7).

Sie können die Verbindungseigenschaften jederzeit über *Daten / Verbindungen* überprüfen und nachbessern. Markieren Sie dazu im Dialog *Arbeitsmappenverbindung* (unter *Daten / Verbindungen*) die entsprechende Verbindung und klicken Sie auf *Eigenschaften*. Auf der ersten Registerkarte *Verwendung* finden Sie die in Bild 8 gezeigten Einstellungsmöglichkeiten (siehe auch Tabelle 1). Die zweite Registerkarte *Definition* enthält u.a. die Verbindungszeichenfolge mit dem Pfad zur Quelldatei. Sollte sich dieser geändert haben, können Sie das hier korrigieren. Andernfalls erhalten Sie beim Versuch, die Daten zu aktualisieren, eine Fehlermeldung.

Strukturierte Verweise in der ODBC-Verbindung

Natürlich können Sie auch eine Tabelle aus verknüpften Daten mit zusätzlichen Spalten versehen und mit strukturierten Verweisen analysieren. Mit der Aktualisierung der Daten aus der ODBC-Datenquelle berechnen sich alle Verweise automatisch neu, und die Formeln, die in zusätzlichen Spalten Berechnungen mit den verknüpften Daten durchführen, kopiert die Tabelle automatisch auf alle Zeilen, die in der aktualisierten Tabelle neu eingefügt wurden.

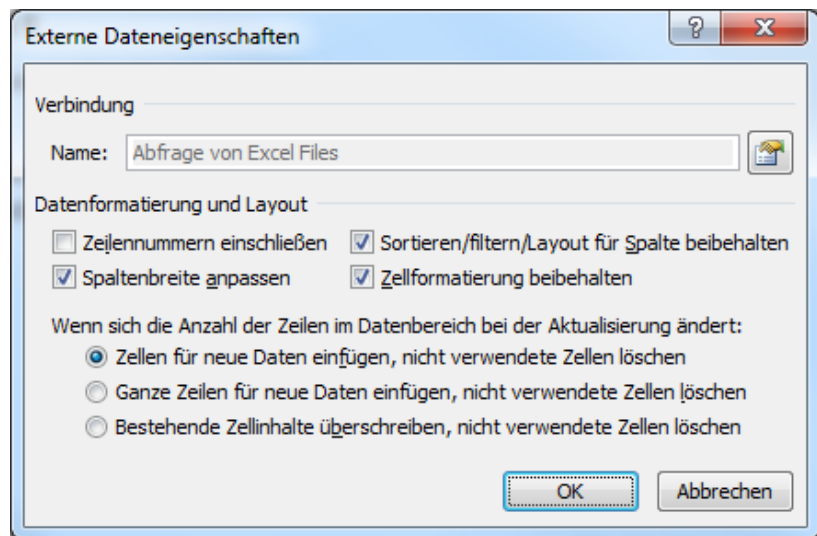


Bild 7: In den Verbindungseigenschaften legen Sie u.a. fest, was passiert, wenn sich die Anzahl der Zeilen im Datenbereich ändert.

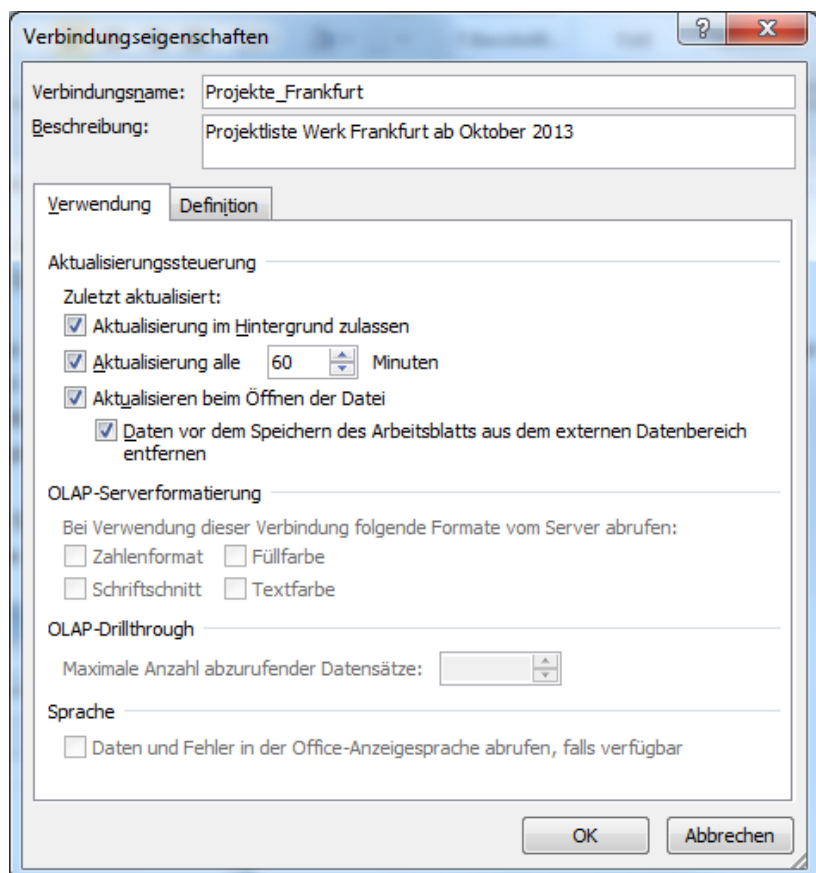


Bild 8: Weitere Verbindungseigenschaften und automatische Aktualisierung der Verbindung.

Verbindungseigenschaften	Bedeutung und empfohlene Einstellung
<i>Verbindungsname und Beschreibung</i>	Tragen Sie hier einen aussagekräftigen Namen und eine Beschreibung für die Verbindung ein
<i>Aktualisieren im Hintergrund zulassen</i>	Wenn Sie das Häkchen entfernen, wird die ODBC-Verbindung deaktiviert.
<i>Aktualisieren alle n Minuten:</i>	Hier können Sie ein Intervall festlegen, in dem die Verbindung automatisch aktualisiert wird. Setzen Sie diese Option, wenn sich die Daten in der Quelldatei häufig ändern.
<i>Aktualisieren beim Öffnen der Datei:</i>	Kreuzen Sie diese Option an, um die erste Aktualisierung schon beim Öffnen der Arbeitsmappe zu starten.
<i>Daten vor dem Speichern des Arbeitsblattes aus dem externen Datenbereich entfernen</i>	Wenn Sie diese zweite Option ebenfalls ankreuzen, enthält Ihre gespeicherte Arbeitsmappe keine Daten, da diese erst mit dem Öffnen der Mappe aus der Quelle abgeholt werden. Diese Option sollten Sie aus Sicherheitsgründen nutzen: Versucht jemand, Ihre Projektübersicht ohne Zugangsberechtigung auf Ihre Datenquellen zu öffnen, wird er keine Daten erhalten.

Tabelle 1: Einstellmöglichkeiten der Verbindungseigenschaften.

Berechnen Sie in unserer ersten Tabelle die Dauer der Projekte und den jeweiligen Fertigstellungswert. Achten Sie darauf, dass die Spalte, die rechts angefügt wird, zur Tabelle gehören muss.

- Markieren Sie mit der rechten Maustaste die Spalte D und fügen Sie über *Zellen einfügen* eine neue Spalte ein.
- Ändern Sie die Spaltenüberschrift von *Spalte D* auf *Dauer*.
- Schreiben Sie die Formel zur Berechnung der Dauer. Geben Sie dabei die strukturierten Verweise ein, indem Sie den Zellzeiger per Cursortasten oder Mauszeiger auf die entsprechenden Werte der angrenzenden Spalten setzen.
$$=[@Ende]-[@Beginn]+1$$
- Formatieren Sie die Spalte D mit dem benutzerdefinierten Zahlenformat:
0" Tage"
- Ziehen Sie das kleine blaue Dreieckssymbol rechts unten mit dem Mauszeiger nach rechts, um die Tabelle um eine Spalte zu erweitern.
- Ändern Sie die Spaltenüberschrift von *Spalte G* auf *Earned Value*.
- Schreiben Sie die Formel zur Berechnung des Fertigstellungswertes mit strukturierten Verweisen auf die Spalten *Fertigstellungsgrad* und *Plankosten*:
$$=[@Fertigstellungsgrad]*[@Plankosten]$$
- Formatieren Sie die Spalten mit dem Zahlenformat der Spalte *Plankosten*.

	A	B	C	D	E	F	G
1	Projekt	Beginn	Ende	Dauer	Plankosten	Fertigstellungsgrad	Earned Value
2	ABM-V Serie VI	15.10.13	14.05.14	212 Tage	320.000	20%	64.000
3	CDLK Serie II	01.10.13	30.04.14	212 Tage	490.000	12%	58.800
4	Kernprozesse optimieren	01.11.13	01.02.14	93 Tage	260.000	0%	0
5	Neubau Lagerhalle Süd	13.11.13	01.10.14	323 Tage	1.500.000	0%	0

Bild 9: Die Projektdauer und der Earned Value werden mit strukturierten Verweisen berechnet.

Mehrere Projektlisten konsolidieren mit SQL

In unserem Beispiel haben wir eine ODBC-Verbindung zu einer der drei externen Tabellen hergestellt. Um eine kompakte Übersicht der Daten aller drei Tabellen zu erstellen, wäre es praktisch, wenn man die Inhalte der beiden anderen Tabellen einfach an die eben erzeugte Tabelle anfügen könnte. Allerdings gibt es in Microsoft Query keine Option, um Daten aus mehreren Tabellen mit gleichem Spaltenaufbau zusammenzufassen. Die Datenbanksprache SQL (structured query language) bietet jedoch eine einfache Möglichkeit, um dies zu erreichen.

Microsoft Query und SQL

Wenn Sie mit Microsoft Query eine ODBC-Verbindung zu einem Excel-Tabellenblatt aufbauen und eine dynamische Verbindung zu einer Liste herstellen, produziert der Assistent eine SQL-Anweisung. SQL ist als Basis für Verbindungen zu externen Daten ideal, weil diese Abfragesprache von fast allen relationalen Datenbanken akzeptiert wird. Wird die Verbindung auf ein Excel-Tabellenblatt bezogen, lässt sich dieser SQL-String sogar anzeigen. Andere ODBC-Verbindungen, zum Beispiel mit Access oder SQL-Server, liefern diese Information nicht. Sehen Sie sich den SQL-Befehl der ersten ODBC-Verbindung an:

- Setzen Sie den Zellzeiger in die Tabelle und wählen Sie *Daten / Verbindungen / Verbindungen*.
- Klicken Sie auf *Eigenschaften*.
- Schalten Sie um auf die Registerkarte *Definition*. Hier sehen Sie unter *Befehlstext* den SQL-Befehl, den Microsoft Query für die ODBC-Verbindung produziert hat (Bild 10).

Die Sprache ist nicht allzu komplex, die wichtigsten Elemente kann der Query-Assistent produzieren. Wer zusätzliche Möglichkeiten von SQL ausschöpfen möchte, muss die Syntax kennenlernen. Tabelle 2 im Anhang zeigt eine Übersicht der wichtigsten Elemente.

UNION konsolidiert Tabellen

Um Tabellen zu konsolidieren, lässt sich das SQL-Statement UNION verwenden. Im Unterschied zu anderen SQL-Elementen wie WHERE (Filter) oder ORDER BY (Sortierung) kann Microsoft Query das Element UNION jedoch nicht produzieren. Sie können den von Microsoft Query erzeugten SQL-Befehl jedoch einfach abändern und um eine zusätzliche Verbindung mit UNION erweitern. Die Feldnamen ersetzen Sie dabei einfach durch einen *, damit ist immer die gesamte Feldliste (Spaltenliste) gemeint.

- Um den SQL-Befehltext anzuzeigen, wählen Sie mit dem Zellzeiger in der Tabelle *Daten / Verbindungen / Verbindungen*.
- Markieren Sie die bestehende Verbindung (Excel Files) und klicken Sie auf *Eigenschaften*. Schalten Sie um auf die Registerkarte *Definition*. Im Feld *Befehltext* sollte der SQL-Befehl, der diese Verbindung aufbaut, wie folgt aussehen (mit Pfad für ihre Pfadangabe zur Quellmappe)

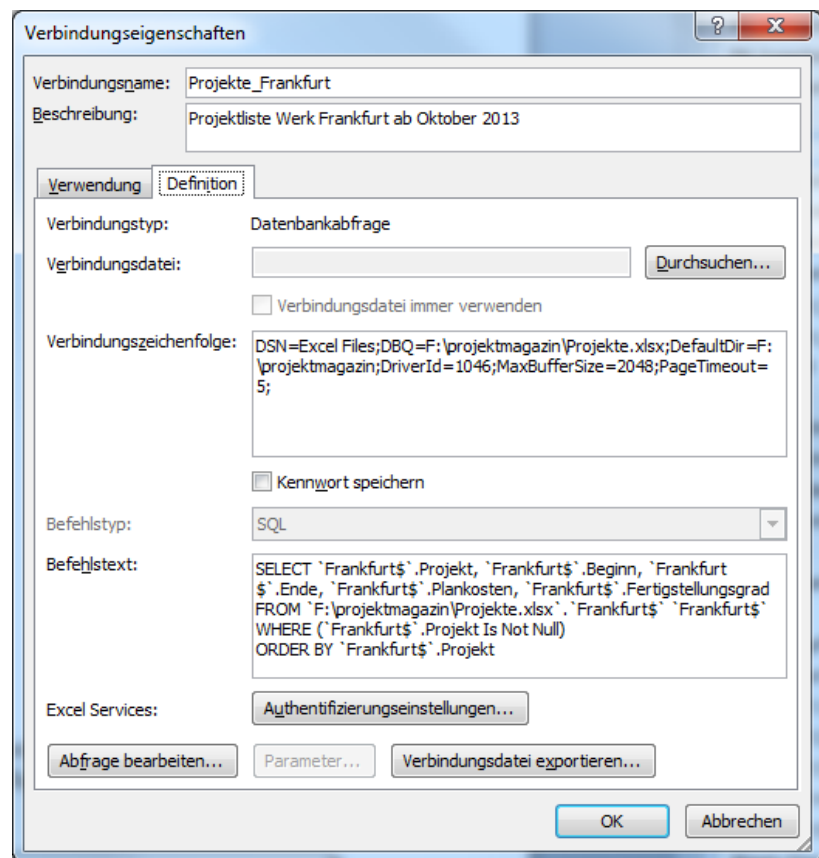


Bild 10: In den Verbindungseigenschaften wird der SQL-String sichtbar.

```
SELECT `Frankfurt$`.Projekt, `Frankfurt$`.Beginn, `Frankfurt$`.Ende, `Frankfurt$`.Plankosten, `Frankfurt$`.Fertigstellungsgrad FROM `Pfad\Projekte.xlsx`.`Frankfurt$` `Frankfurt$` WHERE (`Frankfurt$`.Projekt Is Not Null) ORDER BY `Frankfurt$`.Projekt
```

- Ersetzen Sie in dem Befehltext die Feldnamen durch einen Stern (*). Sie erhalten so automatisch alle Felder der Tabelle. Die Sortierung können Sie löschen, der SQL-String für das erste Tabellenblatt sieht damit wie folgt aus:

```
SELECT * FROM `Pfad\Projekte.xlsx`.`Frankfurt$` `Frankfurt$` WHERE (`Frankfurt$`.Projekt Is Not Null)
```

! Wenn Ihr SQL-Befehl keine Pfadangabe (z. B. C:\Daten\) zur Arbeitsmappe enthält, sucht die Verbindung die Datenquelle im aktuellen Ordner, d.h. in dem Ordner, der bei Excel nach Ausführen von *Datei / Öffnen* angezeigt wird. Um sicher zu gehen, sollten Sie immer den Pfad und den Namen der Arbeitsmappe verwenden.

! Der SQL-Befehl einer ODBC-Verbindung entspricht nicht ganz dem SQL-Standard. In Datenbankabfragen werden Tabellennamen zwischen Apostrophe (') gesetzt, Query verwendet hier Accents (`) und schreibt den Tabellennamen zweimal. Ändern Sie diese Schreibweise nicht und löschen Sie auch keine Teile des aufgezeichneten Strings.

- Erweitern Sie den String um die Anweisung UNION und fügen Sie alle Datensätze aus dem Tabellenblatt *München* hinzu:

```
SELECT * FROM `Pfad\Projekte.xlsx`.`Frankfurt$` `Frankfurt$` WHERE `Frankfurt$`.Projekt Is Not Null
UNION SELECT * FROM `Pfad\Projekte.xlsx`.`München$` `München$` WHERE `München$`.Projekt Is Not Null
```

- Schließen Sie die Verbindung und aktualisieren Sie die Tabelle über *Verbindungen / Alle aktualisieren*. Das Ergebnis ist eine Tabelle mit den konsolidierten Tabellenblätter.

- Ändern Sie Ihren SQL-String noch einmal, fügen Sie auch noch das dritte Tabellenblatt hinzu:

```
SELECT * FROM `Pfad\Projekte.xlsx`.`Frankfurt$` `Frankfurt$` WHERE `Frankfurt$`.Projekt Is Not Null
UNION SELECT * FROM `Pfad\Projekte.xlsx`.`München$` `München$` WHERE `München$`.Projekt Is Not Null UNION SELECT * FROM `Pfad\Projekte.xlsx`.`Stuttgart$` `Stuttgart$` WHERE `Stuttgart$`.Projekt Is Not Null
```

- Schalten Sie nach der Aktualisierung der Tabelle die Ergebniszeile ein und berechnen Sie die Durchschnittsdauer Ihrer Projekte, die Summe der Plankosten, den durchschnittlichen Fertigstellungsgrad und die Summe der Fertigstellungswerte (Earned Values), was Ihren gesamten IST-Kosten entspricht (Bild 11).

	A	B	C	D	E	F	G
1	Projekt	Beginn	Ende	Dauer	Plankosten	Fertigstellungsgrad	Earned Value
2	ABM-V Serie VI	15.10.13	14.05.14	212 Tage	320.000	20%	64.000
3	Ausarbeitung Richtlinien Umweltschutz	15.10.13	22.02.14	131 Tage	160.000	30%	48.000
4	CDK Produktpflege	15.10.13	22.02.14	131 Tage	125.000	40%	50.000
5	CDLK Serie II	01.10.13	30.04.14	212 Tage	490.000	12%	58.800
6	Entwicklung e-commerce-Lösung	01.12.13	15.03.14	105 Tage	260.500	20%	52.100
7	Getriebestrang Automatisierung	01.10.13	31.03.14	182 Tage	90.500	20%	18.100
8	Kapazitätserhebung und Pers.planung	01.11.13	31.03.14	151 Tage	135.000	20%	27.000
9	Karosseriebau Instandhaltung	01.10.13	29.01.14	121 Tage	120.000	80%	96.000
10	Kernprozesse optimieren	01.11.13	01.02.14	93 Tage	260.000	0%	0
11	Neubau Lagerhalle Süd	13.11.13	01.10.14	323 Tage	1.500.000	0%	0
12	Unfallverhütungsvorschriften	15.10.13	15.03.14	152 Tage	56.000	20%	11.200
13	Verbesserung Elektronik	01.10.13	03.02.14	126 Tage	150.000	60%	90.000
14	Ergebnis			1939 Tage	3.667.000	27%	515.200

Bild 11: Die konsolidierte Tabelle mit Ergebniszeile.

Fazit

Tabellen und strukturierte Verweise sind, wie der erste Teil des Beitrags zeigte, richtungsweisende, moderne Werkzeuge der Tabellenkalkulation, die viel Zeit und Arbeit sparen und dem Anwender neue Möglichkeiten für die Aufbereitung und Analyse von Projektlisten aufzeigen. In Verbindung mit der ODBC-Technik bieten Tabellen die ideale Voraussetzung für dynamische Verknüpfungen, Echtzeitverarbeitung von Projektdaten und Konsolidierungen unterschiedlichster Daten. Für einfachere Aufgaben reicht der ODBC-Abfrageassistent Microsoft Query, wer tiefer in die Programmierung von Datenbankabfragen einsteigen will, sollte sich mit SQL beschäftigen. In allen Fällen ist die Verknüpfung der Daten per ODBC und die Weiterverarbeitung der Tabellen mit strukturierten Verweisen sicherer als die alten Copy-and-Paste-Methoden und machen so manche "Formelmonster" mit zahlreichen Verweisen und Verknüpfungen überflüssig.

Hat Ihnen dieser Artikel gefallen?

Bewerten Sie ihn im Projekt Magazin online und teilen Sie so Ihre Meinung anderen Lesern mit. Wählen Sie dazu den Artikel im Internet unter www.projektmagazin.de/ausgaben/2013 oder klicken Sie [hier](#), um direkt zum Artikel zu gelangen.

Anhang – SQL-Statements

SQL-Statement	Erklärung	Beispiel
Select	Wählt einzelne Felder oder alle Felder aus Tabellen	<code>SELECT * FROM Tabelle</code>
Distinct	Wählt in Verbindung mit SELECT nur unterschiedliche Elemente aus Tabellen aus	<code>SELECT DISTINCT * FROM Tabelle</code>
Where	Bedingung für die Auswahl in einer Tabelle	<code>SELECT * FROM Tabelle WHERE Bedingung</code>
And/Or	Logische Begriffe für kombinierte Bedingungen	<code>SELECT * FROM Tabelle WHERE Bedingung1 {[AND OR] Bedingung2 }</code>
In	Auswahlliste für WHERE-Bedingung	<code>SELECT * FROM Tabelle WHERE Feld IN (Wert1, Wert2, ...)</code>
Between	Auswahl mit Unter- und Obergrenze für WHERE-Bedingung	<code>SELECT * FROM Tabelle WHERE Feld BETWEEN Wert1 AND Wert2</code>
Like	Schlüsselwort für genaue Angabe in WHERE-Bedingung	<code>SELECT * FROM Tabelle WHERE Feld LIKE {Wert}</code>
Order By	Gibt Daten auf- oder absteigend sortiert aus	<code>SELECT * FROM Tabelle ORDER BY Feld [ASC, DESC]</code>
Group By	Gruppieren Daten in Verbindung mit einer Funktion nach einem Feld	<code>SELECT SUM(Feld) FROM Tabelle GROUP BY Feld</code>
Having	Filtert die Daten nach Filterkriterien	<code>SELECT * FROM Tabelle GROUP BY Feld HAVING Bedingung</code>
Alias	Verwendet einen Ersatznamen für Spalten, Tabellen oder Funktionen	<code>SELECT Spalte Alias Spaltenalias FROM Tabelle</code>
Create Table	Erstellt eine Tabelle mit Tabellenstruktur	<code>CREATE TABLE Tabelle (Feld1 Datentyp, Feld2 Datentyp,...)</code>
Drop Table	Löscht eine Tabelle inklusive Tabellenstruktur	<code>DROP TABLE Tabelle</code>
Truncate Table	Löscht alle Daten aus einer Tabelle	<code>TRUNCATE TABLE Tabelle</code>
Insert Into	Fügt Daten in eine andere Tabelle ein	<code>INSERT INTO Tabelle Wert1, Wert2, ...)</code>
Update	Aktualisiert Daten in einer Tabelle	<code>UPDATE Tabelle SET Feld = [Wert] WHERE {Bedingung}</code>
Delete From	Löscht Daten über eine Bedingung	<code>DELETE FROM Tabelle WHERE {Bedingung}</code>
Avg, Count, Max, Min, Sum	Funktionen für die Auswahl von Datensätzen	<code>SELECT COUNT(Feld) FROM Tabelle</code>
Join, Inner Join, Outer Join	Verknüpft Tabellen oder Abfrageergebnisse	<code>SELECT * FROM Tabelle INNER JOIN ON Bedingung</code>
Union, Union All	Verbindet Tabellen oder Abfrageergebnisse (ODER-Bedingung)	<code>SELECT * FROM Tabelle1 UNION SELECT * FROM Tabelle2</code>
Intersect	Verbindet Tabelle oder Abfrageergebnisse (UND-Bedingung)	<code>SELECT * FROM Tabelle1 INTERSECT * FROM Tabelle2</code>
Minus	Verbindet Tabelle oder Abfrageergebnisse (NICHT-Bedingung)	<code>SELECT * FROM Tabelle1 MINUS * FROM Tabelle2</code>

Tabelle 2: Die Elemente der Datenbank-Abfragesprache SQL.