Das Verknüpfen von Tabellen in SQL ist einfach – hier erfahren Sie wie es gemacht wird.

von Mike Cravitz

 DB2/400 bzw. UDB/400 (DB2 Universal Database for AS/400) unterstützt Join-Operationen durch verschiedene Mechanismen:

- logische Join-Dateien, mit DDS definiert
- die Parameter FILE und JFLD des Befehls OPNQRYF (Open Query File – Abfragedatei eröffnen)
- die FROM- und WHERE-Klauseln von SQL-Subselects

Von allen Möglichkeiten bietet SQL die größte Flexibilität und die einfachste Syntax. SQL ist auch IBMs strategische Datenbank-Oberfläche für UDB/400, und deshalb ist es sicher sinnvoll, wenn man weiß, wie Dateien mit SQL verknüpft werden. Wie Ihnen dieser Artikel zeigen wird, ist der Einstieg gar nicht so schwierig.

Eine klassische Verknüpfung: Kunden und Umsätze

Sehen wir uns ein klassisches Beispiel für eine Verknüpfung an: Jede Zeile einer Umsatz-Tabelle wird mit der zugehörigen Kundenstamm-Zeile verbunden. Abbildung 1 zeigt die Kundenstamm-Tabelle, Abbildung 2 die Umsatz-Tabelle. Zum Verknüpfen dieser beiden Tabellen verwendet man eine SQL-Anweisung wie die folgende:

> Select OrderID, Customer.CustID, Name From Customer Join Sale On Customer.CustID = Sale.CustID

Das Prädikat, das auf das Schlüsselwort On folgt (Customer.CustID = Sale.CustID) wird als Join-Bedingung bezeichnet. Dieses Beispiel verwendet ein sehr einfaches Subselect – so wird der Teil einer SQL-Anweisung bezeichnet, der mit dem Schlüsselwort SELECT beginnt und eine FROM-Klausel zur Angabe der Tabelle(n) oder Sicht(en) enthält, die die Ergebniszeilen liefern. In unserem Beispiel enthält die FROM-Klausel die SQL-Join-Operation, die die Ergebnistabelle in Abbildung 3 produziert. Einfach, nicht wahr? Bevor wir die Join-Syntax diskutieren, wollen wir kurz erklären, wo Sie Subselects einsetzen können, und wo folglich Joins möglich sind. Zunächst einmal können Sie Select-Anweisungen in IBMs interaktiver SQL-Umgebung oder in jedem anderen Ad-hoc-Tool einsetzen, das SQL unterstützt. Das Ergebnis eines Ad-hoc-Selects kann entweder angezeigt, gedruckt oder in eine Ausgabedatei geschrieben werden. Die Select-Anweisung ermöglicht es sogar, mehrere Subselects mit Union-Operationen zu kombinieren und das Ergebnis mit einer Order-By-Klausel zu sortieren. Eine vollständige Beschreibung der Select-Anweisung wollen wir hier nicht liefern, weil sie für das Verständnis von SQL-Joins nicht erforderlich ist. Sie können eine Select-Anweisung (und damit ein Subselect) auch verwenden, wenn Sie einen SQL-Cursor in einem HLL-

Programm definieren. Ein Cursor, der in eingebettetem SQL oder in der SQL Stored Procedure Language (SPL) verwendet werden kann, erlaubt es, ausgewählte Zeilen einzeln nacheinander zu holen (und gegebenenfalls zu ändern oder zu löschen). Das einzig wichtige, das Sie sich im Zusammenhang mit unserem Thema merken müssen ist, daß ein Cursor für verknüpfte Tabellen immer als read-only (nur Lesen) definiert ist. Sie können auch mit einem Subselect eine Read-Only-Sicht definieren. Subselects können auch an folgenden Stellen erscheinen:

- in einer Insert-Anweisung, die Werte aus einer oder mehreren Tabellen kopiert und zu einer anderen Tabelle hinzufügt
- in der Set-Klausel einer Update-Anweisung, die einer Spalte einen neuen Wert zuweist
- als Teil einer Suchbedingung in der Where-Klausel einer Datenmanipulations-Anweisung

Grundlegende Join-Syntax

Joins können in mehrere Kategorien eingeteilt werden: Wichtigstes Unterscheidungsmerkmal ist die Einteilung in Equijoins und Non-Equijoins. Equijoins sind der gebräuchlichere Typ; sie kombinieren Zeilen basierend auf gleichen Spalteninhalten. Das oben erwähnte Beispiel mit Kunden und Umsätzen ist ein Equijoin. Im Gegensatz dazu basieren Non-Equijoins auf Ungleichheiten zwischen Spalten. Wir wollen uns zunächst mit Equijoins beschäftigen und untergliedern sie weiter in

- Inner Equijoins
- Left Outer Equijoins
- Exception Equijoins

Diese drei Kategorien unterscheiden sich dadurch, wie gleiche und ungleiche Zeilen behandelt weden (siehe Abbildung 4). Abbildung 5 zeigt ein Beispiel für einen Left Outer Join, der aus allen Zeilen eines einfachen Joins und allen Zeilen der ersten (linken) Tabelle (in unserem Fall der Kundentabelle) besteht, für die keine Entsprechung in der zweiten Tabelle gefunden wurde. Wie Sie in der Abbildung sehen können, setzt SQL in allen Zeilen ohne Entsprechung die Spalten der zweiten Tabelle auf Null. Während der einfache Join in Abbildung 3 Informationen über alle Kunden liefert, die einen Auftrag plaziert haben, liefert der Left Outer Join Informationen über alle Kunden, unabhängig davon, ob sie geordert haben oder nicht. Abbildung 6 zeigt ein Beispiel für einen Exception Join, der nur die Zeilen aus der ersten Tabelle enthält, für die keine Entsprechung in der zweiten Tabelle existiert.

Sie müssen sich als Abonnent anmelden um den hier fehlenden Teil des Inhalts zu sehen. Bitte **Login** für Zugriff.

Noch nicht Abonnent? Sonderaktion nutzen.

- 7 Euro/Monat NEWSabo digital - sofort zugreifen & online bezahlen.
- 13,5 Euro/Monat NEWSabo plus inkl. 5x Logins & Print-Ausgaben - sofort zugreifen & per Firmen-Rechnung bezahlen.