Bridge Tables bilden in der dimensionalen Modellierung Dimensionen mit Mehrfach-Attributen (Multi Valued Dimensions) oder rekursive Hierarchien in einer Dimension ab. Diese Erweiterung des Star-Schemas ist zwar mächtig, aber auch komplex in der Anwendung.

Brücken bauen im dimensionalen Modell

Dani Schnider, Trivadis AG

Der Artikel zeigt anhand von konkreten Beispielen, wie Bridge Tables modelliert, geladen und abgefragt werden können, warum Bridge Tables nicht in jedem Fall die beste Lösung sind, wo ihre Risiken liegen und wie diese durch geeignete Alternativen vermieden werden können. Nehmen wir an, die DOAG möchte Auswertungen über die Anzahl von Teilnehmern an den einzelnen Vorträgen an der DOAG-Konferenz machen und erstellt dafür ein Star-Schema mit verschiedenen Dimensionen, unter anderem mit einer Dimension „DIM_SESSION“, in der die verschiedenen Sessions (Vorträge) aufgeführt sind. Die Erstellung eines solchen Star-Schemas stellt kein Problem dar, mit Ausnahme eines kleinen, aber aus Modellierungssicht unschönen Details: Es gibt Vorträge mit mehr als einem Referenten.

Wie immer gibt es mehrere Möglichkeiten, einen solchen Sachverhalt in einem dimensionalen Datenmodell abzubilden. Eine davon ist, die Namen der Referenten als kommaseparierte Liste in einem Attribut abzuspeichern. Diese nicht sehr elegante Lösung ist allerdings schwerfällig für die Abfragen. Andere Varianten sind mehrere Attribute („SPEAKER_1“, „SPEAKER_2“, „SPEAKER_3“) in der Dimensions-Tabelle oder eine separate Dimension „DIM_SPEAKER“, die aus der Fakten-Tabelle mehrfach referenziert wird. Nachteil dieser Lösungen – neben den ebenfalls nicht ganz trivialen Abfragen – ist die Beschränkung auf eine maximale Anzahl von Referenten. Ein pragmatischer Ansatz besteht darin, pro Vortrag einen Haupt-Referenten zu definieren und nur diesen in der Dimensions-Tabelle zu speichern. Diese Lösung ist zwar einfach zu realisieren, führt aber zu fehlenden Informationen bei den Auswertungen.

Multi Valued Bridge Tables

Eine vollständige und einfache Lösung für die Abbildung von Mehrfach-Attributen ist in einem klassischen Star Schema mit Dimensions- und Fakten-Tabellen nicht möglich. Um solche Datenbestände abzubilden, kann jedoch eine weitere Art von Tabellen verwendet werden: die Bridge Table. Wie der Name besagt, bildet diese eine Brücke zwischen zwei Dimensionen oder zwischen einer Dimensions- und einer Fakten-Tabelle. Diese beiden

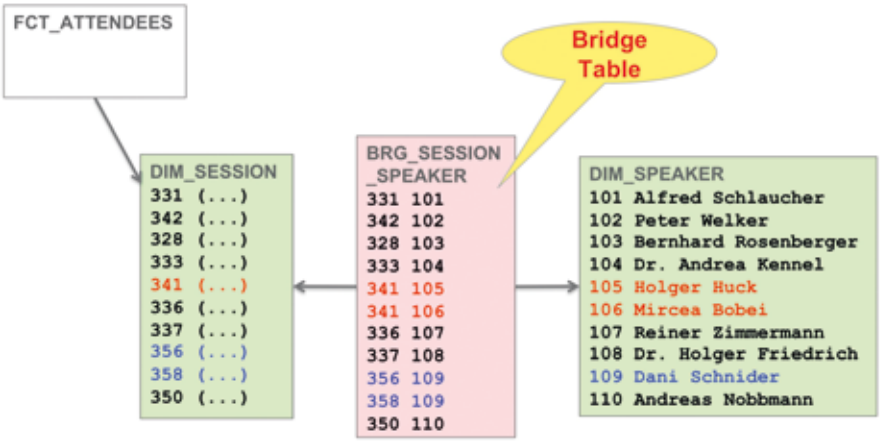

Abbildung 1: Das Beispiel mit Multi Valued Attribute Bridge Table

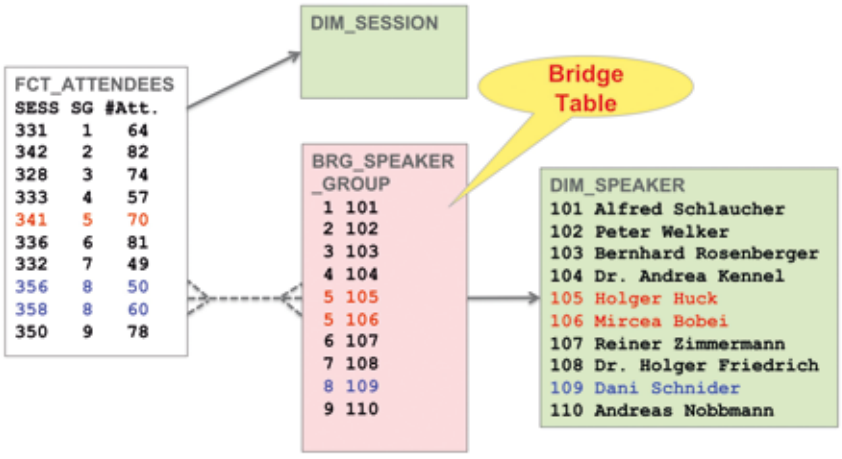

Abbildung 2: Das Beispiel mit Multi Valued Dimension Bridge Table

Möglichkeiten werden anhand unseres Beispiels mit den DOAG-Vorträgen genauer erläutert.

Um das Beispiel zu illustrieren, wurde eine Reihe von Vorträgen aus dem Stream „DWH & BI“ der DOAG-Konferenz 2012 ausgewählt. Die für unser Problem mit den Mehrfach-Attributen interessanteste Session ist dabei der Vortrag „Oracle Essbase Backup & Recovery“. Warum? Weil der Vortrag von zwei Referenten, Holger Huck und Mircea Bobei, gehalten wurde. Um im dimensionalen Datenmodell Sessions mit zwei (oder mehr) Referenten abbilden zu können, wird die Dimension „DIM_SESSION“ durch eine Bridge Table sowie eine zusätzliche Dimensions-Tabelle erweitert (siehe Abbildung 1).

Zusätzlich zur Dimensionstabelle „DIM_SESSION“ wird eine weitere Dimensions-Tabelle „DIM_SPEAKER“ angelegt, in der sämtliche Referenten der DOAG-Konferenz (hier nur ein Ausschnitt) abgespeichert sind. Durch die Bridge Table „BRG_SESSION_SPEAKER“ werden die „n:n“-Beziehungen zwischen „DIM_SESSION“ und „DIM_SPEAKER“ abgebildet, wie wir es aus der relationalen Datenmodellierung kennen. Durch diese sogenannte „Multi Valued Attribute Bridge Table“ lassen sich sowohl Vorträge mit mehreren Referenten als auch Referenten mit mehreren Vorträgen abbilden (Details siehe [1] Seite 205 und [2] Seite 210).

Eine andere Möglichkeit besteht darin, eine Multi Valued Dimension Bridge Table zwischen Fakten- und Dimensions-Tabelle zu verwenden. Dazu ändern wir das Datenmodell unseres Beispiels so, dass die Dimensionen „DIM_SESSION“ und „DIM_SPEAKER“ als unabhängige Dimensionen modelliert und somit separat aus der Faktentabelle referenziert werden. Um Vorträge mit mehreren Referenten abbilden zu können, wird zwischen Fakten- und Dimensions-Tabelle „DIM_SPEAKER“ eine Bridge Table gelegt (siehe Abbildung 2).

Die Einträge in der Fakten-Tabelle „FCT_ATTENDEES“ enthalten die Anzahl der Teilnehmer für die einzelnen Sessions. Anmerkung: Die hier aufgeführten Zahlen sind frei erfunden und entsprechen nicht den tatsächlichen Teilnehmerzahlen. Die Fakten referenzieren jedoch nicht einen einzelnen Referenten in „DIM_SPEAKER“, sondern eine „Speaker Group“ (Attribut „SG“), die in der Bridge Table definiert ist. Auf diese Weise ist es ebenfalls möglich, Vorträge mit beliebig vielen Referenten abzubilden.

Zu beachten ist in diesem Beispiel die „n:n“-Beziehung zwischen Fakten-Tabelle und Bridge Table. Sie verhindert die Definition von Foreign Key Constraints zwischen den Tabellen. Dieses Problem kann jedoch durch eine zusätzliche Dimensions-Tabelle (zum Beispiel „DIM_SPEAKER_GROUP“) mit nur einem Attribut und einem künstlichen Schlüssel gelöst werden, der dann sowohl von der Fakten-Tabelle als auch von der Bridge-Table referenziert wird.

Hohe Flexibilität und hohe Komplexität

Der Vorteil von Bridge Tables liegt in der Flexibilität: Die fachlichen Zusammenhänge mit Mehrfach-Attributen können vollständig abgebildet werden und es gibt keine Limitierung der Anzahl der Werte. Auch ein Vortrag mit zehn oder mehr Referenten könnte in beiden oben erwähnten Daten-Modellen abgebildet werden. Die Flexibilität hat allerdings ihren Preis. Im Falle von Bridge Tables äußert sich dieser durch eine höhere Komplexität, sei es beim Datenmodell („n:n“-Beziehung), in der ETL-Logik oder bei den Abfragen auf das Star-Schema. Hier müssen

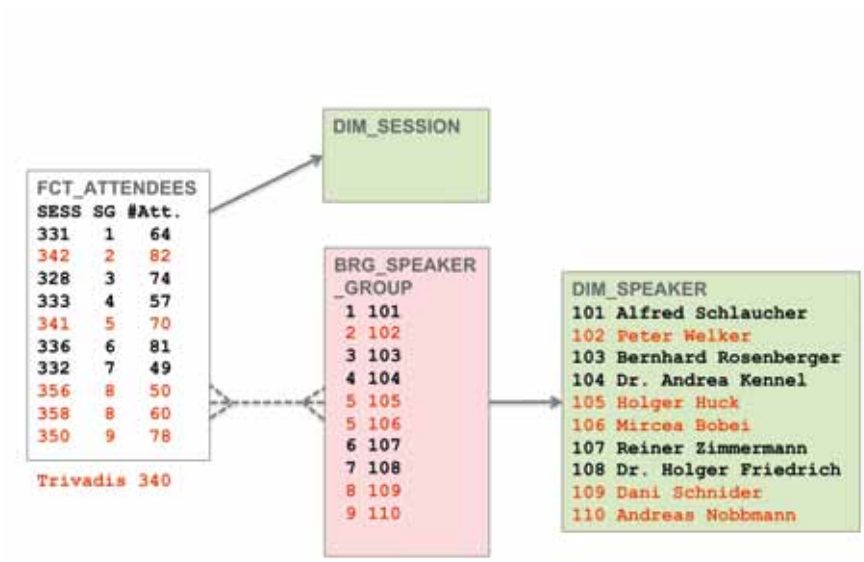

Abbildung 3: Anzahl Vortragsteilnehmer bei Trivadis-Referenten

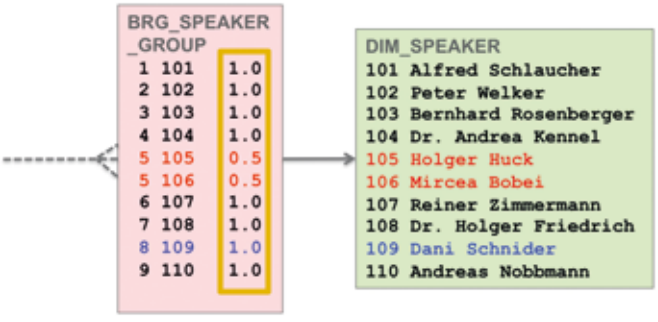

Abbildung 4: Gewichtung der Zuordnungen in der Bridge Table

dann spezielle Vorkehrungen getroffen werden, um Mehrfach-Zählungen zu vermeiden, wie im nächsten Abschnitt beschrieben.

Wo liegt die zusätzliche Komplexität bei den ETL-Prozessen? Neben dem Einfügen oder Ersetzen von Dimensions-Einträgen müssen auch die zugehörigen Datensätze in der Bridge Table bewirtschaftet werden. Das kann zum Beispiel heißen, dass nachträglich ein zusätzlicher Referent für einen bereits angemeldeten und ins DWH geladenen Vortrag angekündigt wird. Dies führt zu einem neuen Eintrag in der Bridge Table. Bei Absage eines Referenten muss die entsprechende Zuordnung aus der Bridge Table gelöscht werden. Lösch-Operationen in einem Data Warehouse gibt es normalerweise nicht – bei Bridge Tables können sie jedoch durchaus zweckmäßig und notwendig sein. Die hier aufgeführten Beispiele gehen von der einfachen Annahme aus, dass keine Historisierung der Dimensionsdaten nötig ist, dass wir es also mit Slowly Changing Dimensions Typ 1 (SCD 1) zu tun haben.

Bei SCD 2 wird es um einiges komplexer. So hat das Einfügen einer neuen Version in die Dimensions-Tabelle auch die Erstellung neuer Versionen aller zugehörigen Einträge in der Bridge Table zur Folge. Eine versionierte Bridge Table wächst dadurch typischerweise sehr rasch, da für jede Änderung eines Dimensionseintrags sämtliche Gruppen-Zugehörigkeiten kopiert werden müssen. Bei Änderungen von Gruppen-Zugehörigkeiten (wie nachträgliche An- und Abmeldungen von Referenten) müssen in der Bridge Table neue Versionen erstellt und teilweise bestehende Einträge kopiert werden. Bei Multi Valued Bridge Tables müssen je nach Art der Änderung auch zusätzliche Versionen in die Dimensions-Tabelle eingefügt werden. Schließlich ist bei Bridge Tables in Kombination mit SCD 2 zu beachteten, dass bei Abfragen immer eine Einschränkung des Datums-Intervalls auf die Bridge Table notwendig ist, da sonst mehrere Versionen aus der Dimensions-Tabelle selektiert werden. Die Einschränkung aufgrund des Joins mit der Fakten-Tabelle, wie sonst bei SCD2-Dimensionen üblich, genügt hier nicht.

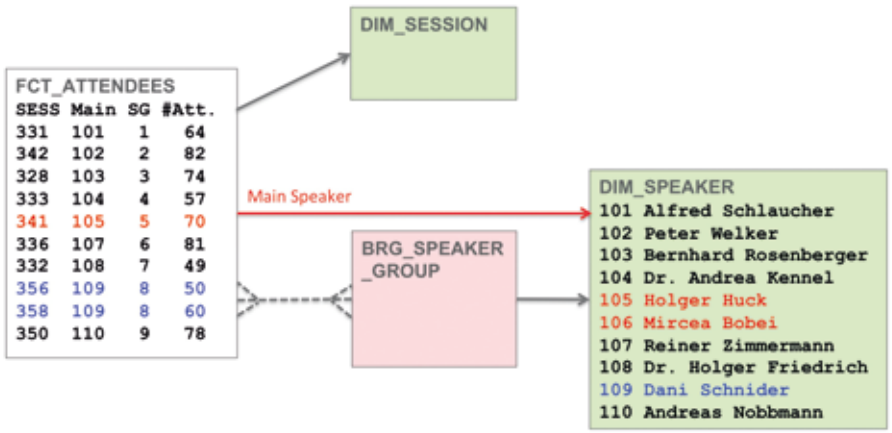

Abbildung 5: Vereinfachung durch View über Bridge Table

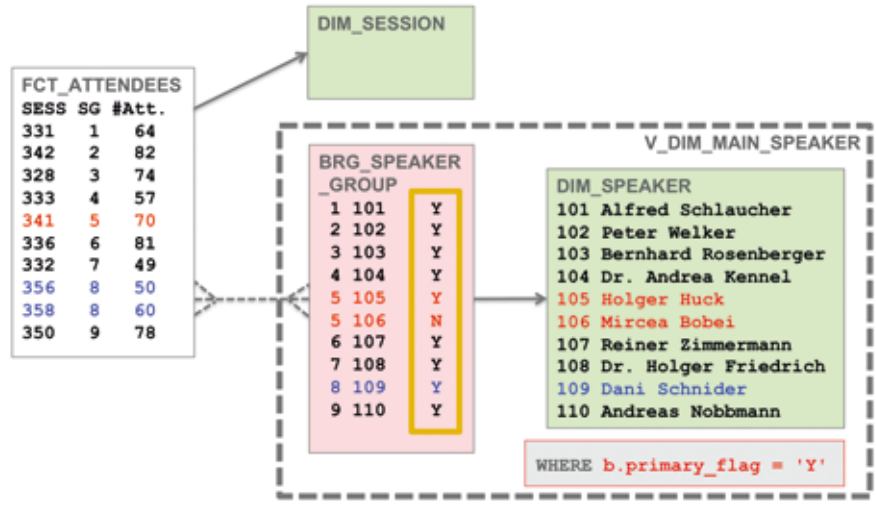

Abbildung 6: Vereinfachung durch zusätzliche Beziehung auf Dimensionstabelle

Abfragen auf Bridge Tables

Der letzte erwähnte Punkt führt uns zu einer wesentlichen Fehlerquelle im Zusammenhang mit Bridge Tables: Mehrfachzählungen bei Abfragen. Um die Problematik zu erläutern, führen wir ein paar SQL-Abfragen auf das Beispielschema aus Abbildung 2 aus. Zuerst möchten wir wissen, wie viele Teilnehmer jeder Referent in seinen Vorträgen hat (siehe Listing 1).

```
SELECT d.speaker_name
     , SUM(f.num_attendees)
  FROM fct_attendees f
  JOIN brg_speaker_group b
ON (b.speaker_group_id =
f.speaker_group_id)
  JOIN dim_speaker d ON
(d.speaker_id = b.speaker_id)
GROUP BY d.speaker_name
```

Listing 1

Die Query liefert für alle Referenten korrekte Resultate. Dass Holger Huck und Mircea Bobei je 70 Zuhörer haben, liegt daran, dass sie einen gemeinsamen Vortrag präsentieren. Aus Sicht jedes einzelnen Referenten ist die ermittelte Anzahl der Teilnehmer korrekt.

Nun möchten wir die Abfrage so ändern, dass die Anzahl der Teilnehmer nicht pro einzelnen Referenten, son-

dern pro Firma, bei der die Referenten angestellt sind, ermittelt wird. Dieser „Drill-Up“ wird üblicherweise so realisiert, dass einfach nach einem anderen Attribut der Dimension – hier nach dem Firmennamen – aggregiert wird (siehe Listing 2).

```
SELECT d.company_name
     , SUM(f.num_attendees)
  FROM fct_attendees f
  JOIN brg_speaker_group b
ON (b.speaker_group_id =
f.speaker_group_id)
  JOIN dim_speaker d ON
(d.speaker_id = b.speaker_id)
GROUP BY d.company_name
```

Listing 2

Doch liefert diese SQL-Abfrage das korrekte Resultat? In der für das Beispiel willkürlich zusammengestellten Liste von Referenten sind „zufälligerweise“ die Hälfte der Personen Trivadis-Mitarbeiter (siehe Abbildung 3).

Werden die (erfundenen) Teilnehmerzahlen der fünf Trivadis-Vorträge zusammengezählt, ergibt die Summe 340 Teilnehmer. Die SQL-Query gibt jedoch als Resultat die Zahl 410 zurück. Wo liegt der Fehler?

Die Ursache liegt bei der Doppelzählung der 70 Teilnehmer, die dem Vortrag von Holger Huck und Mircea Bobei folgen. Da dieser Vortrag von zwei Referenten gehalten wird, ergibt die SQL-Query für diesen Vortrag die doppelte Anzahl an Teilnehmern – also 70 zu viel.

Zur Vermeidung von Mehrfachzählungen wird in der Bridge Table ein zusätzliches Attribut mit einer Gewichtung eingeführt (siehe Abbildung 4). Vorträge mit einem Referenten erhalten die Gewichtung 100 Prozent (beziehungsweise 1.0), bei Vorträgen mit mehreren Referenten wird die Gewichtung prozentual auf die Referenten verteilt – bei zwei Referenten also je 50 Prozent (beziehungsweise 0.5).

Diese Gewichtung wird für die Korrektur von Mehrfachzählungen bei Abfragen auf übergeordnete Aggregationsstufen (wie Referenten einer Firma, eines Landes oder für das Gesamttotal) verwendet (siehe Listing 3). Aber aufgepasst: Bei Abfragen auf der untersten Stufe (Teilnehmerzahl pro Referent) darf die Gewichtung nicht verwendet werden.

```
SELECT d.company_name
     , SUM(f.num_attendees *
b.allocation_factor)
  FROM fct_attendees f
  JOIN brg_speaker_group b
ON (b.speaker_group_id =
f.speaker_group_id)
  JOIN dim_speaker d ON
(d.speaker_id = b.speaker_id)
GROUP BY d.company_name
```

Listing 3

Abbildung 7: Dimensions-Tabelle mit rekursiver Hierarchie

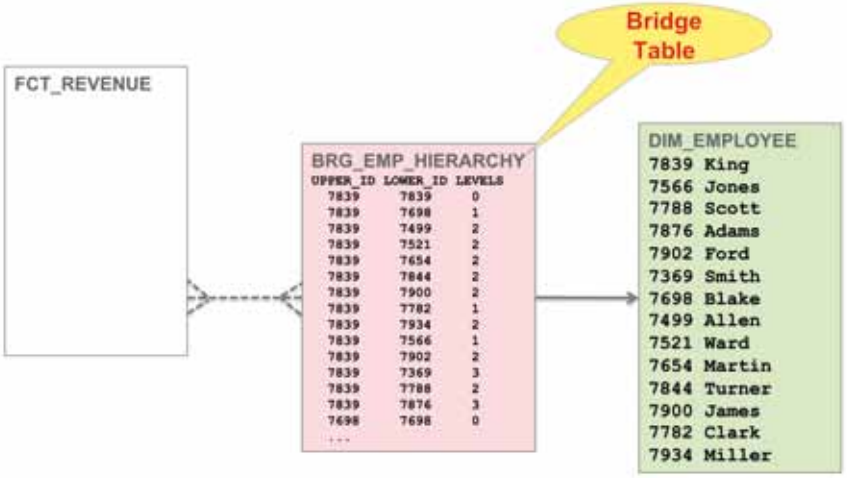

Abbildung 8: Beispiel mit Hierarchy Bridge Table

Vereinfachung der Abfragen

Einmal mehr zeigt sich hier das Dilemma zwischen Flexibilität und Komplexität. Für erfahrene Power-User, die unterschiedlichste Auswertungen nach verschiedenen Kriterien durchführen möchten und in der Lage sind, entsprechende Ad-hoc-Queries zu formulieren, bietet ein Datenmodell mit Bridge Tables zahlreiche Möglichkeiten. Doch die meisten Endanwender – und viele BI-Tools – scheitern an der Komplexität der Abfragen. Hier sind Vereinfachungen gefragt.

Eine Möglichkeit zur Vereinfachung besteht darin, die Komplexität der Bridge Table hinter einer View zu verstecken. Dazu wird die Bridge Table um ein zusätzliches Attribut „PRIMARY_FLAG“ ergänzt. Für jede Referenten-Gruppe ist eine Person als Haupt-Referent markiert. Die View schränkt nun den Datenbestand so ein, dass pro Vortrag nur der jeweilige Hauptreferent angezeigt wird (siehe Abbildung 5). Die meisten Endanwender arbeiten mit dieser View wie mit einer „normalen“ Dimensions-Tabelle. Für spezielle Auswertungen, in denen auch die zusätzlichen

Referenten gefragt sind, wird hingegen direkt auf die Bridge Table und die zugehörige Dimensions-Tabelle zugegriffen.

Als weitere Variante kann eine zusätzliche Beziehung zwischen Fakten- und Dimensions-Tabelle definiert werden, die den Haupt-Referenten jedes Vortrags identifiziert (siehe Abbildung 6). Die Standard-Abfragen der Endanwender verwenden ausschließlich diese Verbindung zur Dimensions-Tabelle „DIM_SPEAKER“, während die Bridge Table nur für spezifische Abfragen durch entsprechend geschulte Power-User zur Anwendung kommt.

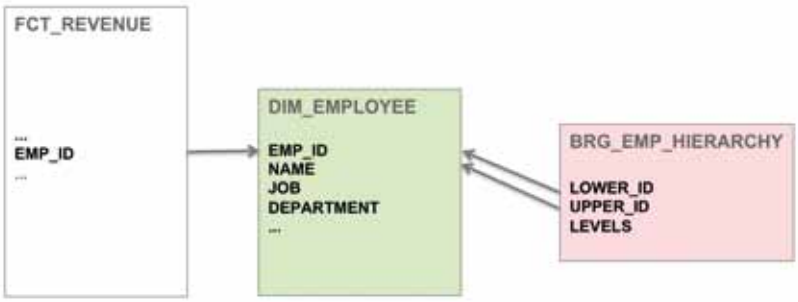

Abbildung 9: Eliminierung der „n:n“-Beziehung einer Hierarchy Bridge Table

Rekursive Hierarchien

Wir haben uns nun ausführlich mit einem Einsatzgebiet von Bridge Tables befasst, nämlich mit der Abbildung von Mehrfach-Attributen in Dimensionen. Daneben gibt es aber noch einen weiteren typischen Anwendungsbereich: rekursive Hierarchien, wie sie zum Beispiel in Mitarbeiter-Organigrammen, Organisationseinheiten, Stücklisten oder Kostenstellen zum Einsatz kommen. Eine rekursive Hierarchie besteht aus Dimensions-Einträgen, die auf übergeordnete Dimensions-Einträge (wie den Vorgesetzten eines Mitarbeiters) verweisen.

Typisch für solche Hierarchien ist, dass die Anzahl der Hierarchie-Stufen nicht fix ist. Eine flexible Möglichkeit besteht in der Implementierung mittels Self-Relationship (auch „Schweinsohr“ genannt), also einer Fremdschlüssel-Beziehung auf die gleiche Tabelle (siehe Abbildung 7). In Oracle SQL lassen sich darauf hierarchische Abfragen ausführen (siehe Listing 4). Neben der Einschränkung, dass diese Abfrage Oracle-spezifisch ist, besteht auch der Nachteil, dass solche Abfragen in vielen BI-Tools nicht oder nur mit erheblichem Aufwand realisiert werden können.

Ein häufig gewählter und bewährter Ansatz besteht darin, die rekursive Hierarchie als flache Dimensions-Tabelle zu implementieren und fehlende Hierarchiestufen durch Wiederholung der übergeordneten Einträge zu füllen (siehe [2] Seiten 224 – 227). In vielen Fällen ist diese Lösung zweckmäßig, hat allerdings die Eigenschaft, dass die Anzahl der Hierarchie-Stufen durch das Design der Dimensions-Tabelle beschränkt wird. Falls diese Einschränkung ein Problem darstellen sollte, lässt sich eine rekursive Hierarchie auch mit einer Bridge Table abbilden.

Hierarchy Bridge Tables

Eine Hierarchy Bridge Table ist eine Tabelle, die für jede Kombination von Dimensions-Einträgen eine Referenz auf den übergeordneten und den untergeordneten Datensatz sowie auf die Anzahl der Hierarchie-Stufen dazwischen festhält. Das Beispiel in Abbildung 8 zeigt eine Mitarbeiter-Dimension, die die 14 Mitarbeiter der altbekannten EMP-Tabelle aus dem Oracle-Beispielschema „SCOTT“ enthält. Um die gesamte Mitarbeiter-Hierarchie abzubilden, sind in der zugehörigen Bridge Table 39 Einträge erforderlich, die nicht alle hier dargestellt sind. Soll nun zum Beispiel der Umsatz aller Mitarbeiter ermittelt werden, die Mr. Jones unterstellt sind, lässt sich dies mit einer einfachen SQL-Abfrage formulieren (siehe Listing 5).

Durch Vertauschen der Attribute „LOWER_ID“ und „UPPER_ID“ der Bridge Table lassen sich auch ähnliche Abfragen formulieren, die die übergeordneten Datensätze aufsummieren (etwa „Mr. Jones und alle seine Vorgesetzten“). Wie Abbildung 8 zeigt, gibt es zwischen der Fakten-Tabelle und der Bridge Table wiederum eine „n:n“-Beziehung. Bei einer Hierarchy Bridge Table kann diese auf einfache Weise eliminiert werden, indem das Datenmodell wie in Abbildung 9 modelliert wird.

Literatur

[1] Ralph Kimball, Margy Ross: The Data Warehouse Toolkit, Second Edition John Wiley and Sons, Inc., 2002, ISBN 978-0471200246

[2] Christopher Adamson: Star Schema, The Complete Reference McGraw-Hill Companies, 2010, ISBN 978-0071744324

Dani Schnider
dani.schnider@trivadis.com

```
SELECT emp_id, name, parent_
emp_id
  FROM dim_employee
 START WITH name = 'Jones'
 CONNECT BY PRIOR emp_id =
parent_emp_id
```

Listing 4

```
SELECT SUM(f.amount)
  FROM fct_revenue f
  JOIN brg_emp_hierarchy b ON
(b.lower_id = f.emp_id)
  JOIN dim_employee d ON
(d.emp_id = b.upper_id)
 WHERE d.name = 'Jones'
```

Listing 5