The Data Warehouse Toolkit: 12章

久野　靖*

2004.6.8

1 Education

□ 教育機関を考える

□ 受験生パイプライン→累積スナップショットとして扱う

- 5章でも累積スナップショットは扱ったが…
- ここでは受験生がマイルストンを通過してく様子をモニタ

□ factless fact tableの使用例

2 University Case Study

□ 大学その他の教育機関に勤めているものとしよう（実際勤めているけど…）

- これからは大学も投資/回収をきっちり考えないと…（耳が痛い）
- 増大するコスト、他大学との競争に耐えて「高品質の学生」を誘引し確保しないといけないのだ!!!
- 卒業したら終わりではない
- お客様（学生さん）はどのような商品（科目）を購入しているだろうか…
- 高いコストの掛かっている教員を目一杯活用する!!!（恐ろしい…）

3 Accumulating Snapshot for Admissions Tracking

□ 5章では注文完遂に至る累積スナップショットを扱った。また特定生産品を累積スナップショットで追跡することも扱った。累積スナップショットの特性は:

- 1つの行が「あるもの」の完全な履歴を表す
- 比較的存在期間の短い「もの」に適する
- マイルストンに対応して複数の日付が記録される
- open-endedなfact集合→さまざまな計測を記録
- 「何か」が起きるごとにその「もの」の行を書き換え
- 書き換え→FKや計測値の変更

□ 受験生の入試プロセスを累積スナップショットで扱う

- 他の業界でも求職者とか売り込み相手とか類似

□ 受験生の追跡の場合、多数のマイルストンを順次通過していく

- どの段階で何人いるといった情報が必要
- 漏斗のように先にいくほど候補者が絞られていく

□ 粒度→1人の応募者について1行。先に進むほど行に多くの情報が記入されていく。(図12.1)

□ 1つの行に複数の日付を記入→ role playing dimension

□ 応募者dimension→多くの興味深い属性

- 地域、成績情報、出身校など→分析対象→次の段に進む学生を絞る

□ 短い間存在する「もの」の追跡→累積スナップショットが適する

- 最新の情報が参照できる
- 「予想合格率」なんかも?→翌年の学生層の予測が可能

4 Factless Fact Tables

□ これまでのところfact tableの構造はどれも類似

- 3〜15くらいのFKカラム
- 1〜数十くらいの数値データ（できればadditive）
- →dimension key値（群）に係わる計測値を保持していると考えられる

□ ここで、もっと別種のfact tableを提示→「数値がない!」→factless fact table

*筑波大学大学院経営システム科学専攻

- ここではイベント追跡用とカバレッジ用のものの例
- これまでにも 2 章と 5 章でカバレッジ型のものは出て来た

4.1 Student Registration Events

□ 複数のディメンション群の「遭遇」イベントを記録することはよくある

- たとえば各学期における学生の科目登録
- fact table の 1 行→「あるコースに、ある学期に、ある学生が登録」(図 12.2)

□ fact data は学期レベル (日付レベルでなく)

- だが日付次元に適合している必要はある
- つまり日付はその日が属する学期 (2002 年度秋学期)、年度 (2002 年度)、学期の季節 (秋学期) などの属性を持つ必要、ラベル等も適合

□ 学生次元は先の応募者次元の拡張版

- 受験時の情報はすべて残す
- 加えて入学後の情報→ 1 部/2 部、居住地、運動、専攻、学年等
- 6 章で述べたのと同様、いくつかのものはミニ次元化←変化していく

□ fact table はこれら次元間の多対多の関係をうまく表現する手段

- 時空間内でこれらの次元の「接点」を記録

□ 多数の興味深い問い合わせに応えられる

- 例: どの学生がどのコースに登録?
- 例: 工学専攻学生の何人が専攻外のファイナンスコースに登録?
- 例: 過去 3 年にある教官の科目に登録した学生は何人?
- 例: その教官の科目を 2 つ以上取っているのは何人?

□ fact には数値はないから、データは「個数」ということに

□ SQL で factless fact table から個数を検索する→非対称なクエリー

```
SELECT FACULTY, COUNT(TERM_KEY) ... GROUP BY FACULTY
```

- TERM_KEY の個数を数えても「学生が何人」が分かるところがちょっとヘンに思えるけどこれが SQL というもの
- 本当に「学期がいくつ」を問いたければ COUNT DISTINCT を使う

□ とはいえ、分かりにくいので「常に 1 という値を持つ」「登録数」とかいうフィールドを作るといいかも (「ダミー」はよくない)

```
SELECT FACULTY, SUM(REGISTRATION_COUNT) ... GROUP BY FACULTY
```

- こうなると厳密には factless ではないようだが、「常に 1」なんだから実質 factless
- 集計したサマリーテーブルを作るなら「登録数」はそこでは意味のあるデータになる

□ 設計していくうちに、実際に意味のあるデータが現れることもある (ただし学期単位という grain に適合していること!)

- たとえば授業料収入、履修時間、成績 → 当然 factless ではなくなる

4.2 Facilities Utilization Coverage

□ factless fact table の 2 番目のタイプ→カバレジ表

□ 例:大学では施設に多くのコストが掛かっている→どの施設がどれくらい働いているか検討したい

- 例:最も多く活用されている (ふさがっている) 施設はどれか?
- 例:各時間帯ごとに、最も多く使われている施設はどれか?
- 例:金曜日には使用率が下がるか? (誰も金曜日には授業出たくない)

□ これらのデータのためにも factless fact table が活用できる

- 各施設、各時間帯 (時限) ごとに 1 行ずつを (施設の利用有無に関わらず) 入れる→図 12.3

□ 施設次元はその施設の記述を全部入れる。建物、施設種別、広さ、定員、アメニティ

□ 利用次元は「利用可能」「使用中」

- 複数の組織が関わって来る場合も

4.3 Student Attendance Events

□ 学生のコースへの出席も同様のスキーマで表現可能→図 12.4

- この場合は各学生×科目×各開講日ということに (学期 vs 日の違い)
- 科目登録 fact と共通する次元が多数ある

□ クエリーの例: 「もっとも出席が多かったコースは?」「学期全体を通じて最も出席が少かったコースは?」「どの教官が最も多くの学生を教えたか?」

4.4 Explicit Rows for What Didn't Happen

□ 「コースに登録したけど出席しなかった」も知りたいことがら

- 「欠席」イベントを登録することも考えられる
- 「欠席」は「出席」と同じ次元群を持つからこれが可能
- 表が大きくなりすぎることもない
- 「出席」fact → 0 または 1 →これを入れるた fact table は factless でなくなる

□ 一般の場合は「不在」を fact に入れるのはまずいことが多い

4.5 Other Relational Options for What Didn't Happen

□ 多くのケースではトランザクションは「疎」となる

- たとえば、それぞれの日においてすべての商品のうち売れるのはごく一部
- このため、「売れなかった」トランザクションを入れると表が大きくなりすぎる
- 2 章の「宣伝カバレジ fact table」のような方法で「宣伝したが売れなかったもの」を分析可能←宣伝された品文句に限定されるため
- このような特定テーブルでは (宣伝が週単位であれば) 粒度を日でなく週にするなどの方法も可能
- 売れた製品については売れたデータから取って来て突き合わせ可能

□ もう 1 つの方法として、SQL の「NOT EXIST」機能を使うことも

- 「ない」データを用意する必要がないのでよさそうだが欠点も
- NOT EXIST を入れ子クエリとしてきっちり区切って呼ぶ必要がある

□ 例: 宣伝したけど売れなかった製品を知りたいとすると…

- まず当該期間において売れたすべての製品を列挙し、その範囲内で NOT EXIST をサブクエリーとして使用する
- 弱点: 「まったく売れなかった」商品をカバーし損なう
- 弱点: 遅い (複雑なクエリだから)
- 弱点: データアクセスツールによっては対応していないかも

□ 図 12.5 のテーブルについて、San Antonio Main Outlet において、2002.1 月に販売されたが、2002.1.15 の ActivePromotion で宣伝したのに売れなかった商品を取り出す SQL

```
SELECT P1.PRODUCT_DESCRIPTION
FROM SALES_FACT F1, PRODUCT P1, STORE S1,
 DATE D1, PROMOTION R1
WHERE F1.PROD_KEY = P1.PROD_KEY
 AND F1.STORE_KEY = S1.STORE_KEY
 AND F1.DATE_KEY = D1.DATE_KEY
 AND F1.PROMO_KEY = R1.PROMO_KEY
 AND S1.STORE_LOCATION = 'San Anto...'
 AND D1.MONTH = 'January, 2002'
 AND NOT EXISTS
 (SELECT R2.PROMO_KEY
  FROM SALES_FACT F2, PROMOTION R2, DATE D2
  WHERE F2.PROMO_KEY = R2.PROMO_KEY
   AND F2.PROD_KEY = F1.PROD_KEY
   AND F2.STORE_KEY = F1.STORE_KEY
   AND F2.DATE_KEY = D2.DATE_KEY
   AND R2.PROMOTION_TYPE = 'ActivePromotion'
   AND D2.FULL_DATE = 'January 15, 2002')
```

4.6 Multidimensional Handling of What Didn't Happen

□ OLAP ツールであれば「ないもの」をうまく扱ってくれる

- データキューブ構築時に「疎な」データに対処→「0」が多数でも OK
- fact が疎すぎなければ「ある」「ない」をともに扱える→関係データベースだけでやるより楽に

5 Other Areas of Analytic Interest

☐ 閑話休題。(tangentってそういう意味だったの?)

☐ 高等教育機関の話に戻ると…

- 前章までで扱った購買、HRM などはすべて大学でも役に立つ
- 収入側はほかに研究資金、同窓生の寄付なども

☐ 研究資金の分析は 7 章の財務分析と似ているがより細かく見ると subledger(その部分についてのみの元帳?) のようなもの

- 粒度としては資金に関するものを追加 (企業提供/政府資金、研究対象、期間、研究者)
- 研究プロジェクトごとにその予算と執行を把握したいというニーズ
- 経費を最小化し、資金を有効活用
- 多くの次元ごとに累計し分析→適切な運営かどうかチェック

☐ 同窓生の寄付は CRM によく似ている

- 同窓生の情報→所在、職、興味、その他の情報 + 生徒として在籍時の情報
- これらの情報を活用→同窓生をよりうまく活用

☐ 寄付だけでなく、職の斡旋、仕事発注、共同研究などでも活用

- フルスケールの CRM システムによってすべての接点を記録することも

6 Summary

☐ accumlating snapshot

- トランザクションや定期スナップショットに比べて使われることが少いが…
- 比較的短期間で終わるプロセスでマイルストンが標準化されているものに適用すると好都合

☐ faltless fact table

- 複数次元間の「関連」を捕捉
- 「起きなかったこと」を扱うための工夫