A TSQL2 Tutorial*

Richard T. Snodgrass Ilsoo Ahn Gad Ariav Don Batory James Clifford
Curtis E. Dyreson Ramez Elmasri Fabio Grandi Christian S. Jensen Wolfgang Käfer
Nick Kline Krishna Kulkarni T. Y. Cliff Leung Nikos Lorentzos John F. Roddick
Arie Segev Michael D. Soo Suryanarayana M. Sripada

1 Introduction

This tutorial presents the primary constructs of the consensus temporal query language TSQL2 via a media planning scenario. *Media planning* is a series of decisions involved in the delivery of a promotional message via mass media.

We will follow the planning of a particular advertising campaign. We introduce the scenario by identifying the marketing objective. The media plan involves placing commercials, and is recorded in a temporal database. The media plan must then be evaluated; we show how TSQL2 can be used to derive information from the stored data. We then give examples that illustrate storing and querying indeterminate information, comparing multiple versions of the media plan, accommodating changes to the schema, and vacuuming a temporal database of old data.

2 The Context

When Miller Brewing Company introduced Lite Beer, it revolutionized the beer industry. Once thought of as weak and feminine, low-calorie beer became the drink of choice for many consumers in the most lucrative market—males who drink large quantities of beer[1]. Not to miss a profitable opportunity, Anheuser-Busch introduced Bud Light, which had fewer calories than its regular beer, Budweiser. The new beer was intended to compete with Lite Beer from Miller[2].

From a promotional standpoint, there were two major tasks: to link the new brand to the established image of its flagship brand, Budweiser, and to differentiate Bud Light from Lite Beer from Miller.

Anheuser-Busch was committed to giving Bud Light the support necessary to become a leading contender in the light beer category. But to compete with the phenomenonally successful Lite Beer, it would be necessary to achieve high brand awareness, and quickly. This would require the promotional media plan to achieve broad reach and high impact[3].

Commercials during the Super Bowl (the professional U.S. football championship game) met these requirements admirably. The Super Bowl is among the most watched television events of each year[4].

To ensure high impact, the commercials mirror the football game. A team composed of bottles of Budweiser "competes" against a team composed of bottles of Bud Light, with one commercial each quarter of the game. It was hoped that the series of four commercials would be highly memorable and generate high excitement, and would help to link the Bud Light brand to Budweiser in the consumers' minds. Another benefit was its potential for promotional tie-ins.

Anheuser-Busch faced the challenge of integrating the Super Bowl commercials for Bud Light with ongoing promotion for both Budweiser and for Bud Light, which involved heavy TV coverage. This would involve generating a comprehensive media plan.

A *media plan* is a set of specific media objectives (in this case, increase brand awareness of Bud Light), and specific media strategies that determine which spots

*Correspondence may be directed to the following address: R. T. Snodgrass, University of Arizona, Computer Science Dept., Tucson, AZ 85721, rts@cs.arizona.edu.

[1] Miller's commercials, involving various sports celebrities arguing "Less Filling" versus "Tastes Great," were critical in achieving this positioning.

[2] You may recall the TV commercials. A person walks up to a bar and asks for a "light." Instead of getting a beer, he gets a spotlight, or a candle, or a match, making a pun on the word "light." In another series of commercials, the original 'party animal,' Spuds MacKenzie, was created to personify the attitude of the Bud Light drinker.

[3] *Reach* is the percentage of the target market that is exposed to the message. A broad reach is a high reach across a large, undifferentiated target market.

[4] The Super Bowl has been the occasion for several highly memorable ads, which many believe was initiated by Apple Corporation's "1984" ad on the Apple II computer, recently voted the best ad of the decade, and the subsequent "Lemmings" commercial of 1985 introducing the Apple MacIntosh computer. The MasterLock ad in which a rifle bullet is shot into a lock is aired only one time a year, during the Super Bowl, yet many people still recall seeing that ad.

ShowName	*InsertionLength*	*Cost*	*Valid Time*
'Roseanne'	'30' SECOND	251000	'Spring Season 1994'
'Super Bowl'	'60' SECOND	1800000	'Spring Season 1994'

Figure 1: Two Rows of the *NBCShows* Table

`InsertionWindow` specifies which quarter the commercial is to appear in, and is relative to the start of the game. For this, we use another user-defined granularity, specific to the kind of game, rather than the network (there might be other granularities for games with halves, such as basketball, or games with commercials only at the half, such as soccer). Here, the `InsertionLength` has an underlying granularity of `SECOND`, and a range of $10^3 = 1000$ seconds.

The `AS` clause indicates that this is a *bitemporal event table*, with both valid-time support (the timestamp indicates which day the show airs) and transaction-time support. The `EVENT` reserved word indicates that rows are timestamped with sets of datetimes (specifically, `DATE`s), rather than periods, as in the previous example. Here the timestamps are to the underlying granularity of `HOUR`, with a range of 100 years (requiring only one 32-bit word). The *transaction time* is the time the fact is stored in the database. In this case, the table supports multiple versions of the media plan. The DBMS supplies the range and underlying granularities for transaction timestamps.

Anheuser-Busch purchased four one-minute commercials for the 1994 Super Bowl, at $1,800,000 each, as well as a short commercial for Bud Ice Draft, for a total cost of $7,650,000, see Figure 2.

The `football` calendar interprets the literal `INTERVAL 'Second Quarter'`; the datetime literal `'1994-01-30 13'` is interpreted by the `SQL92` calendar. The DBMS supplies the transaction time when rows are inserted or updated. The transaction time is not shown in the table, due to lack of space.

4 Evaluating the Media Plan

Once a media plan is in place, it must be evaluated. We provide some representative queries, chosen to illustrate novel features of TSQL2 which are useful in this application.

One question that arises is how well the media plans for the two beer brands have been integrated. Too many commercials in a single show does not increase the effective reach (although it does increase frequency, which is linked to memorability).

Example. *List those football games broadcast by NBC that have two or more commercials.*

```
SELECT N.GameName
FROM NBC_FB_Insertion AS N N2
WHERE N.GameName = N2.GameName AND
    N.CommercialID <> N2.CommercialID
```

While this appears to be a straightforward SQL-92 query, in fact it is a temporal query, evaluated over a bitemporal event relation, with the result being a valid-time event relation.

Two correlation names are defined, `N` and `N2`, and an implicit join is requested. Since the transaction time is not mentioned in the where clause, the information as best known now is retrieved. Erroneously stored information that was later corrected will not be retrieved.

The query also performs an implicit *valid-time selection*, requiring that rows associated with `N` and `N2` be valid at the same time.

Finally, the query performs an implicit *valid-time projection*. The games that are returned will be associated with particular days, those days in which there were `N` and `N2` rows that were concurrently valid. □

We might want to advertise on long-running shows. Hence, we may want to know how long a show has been running. This information can be extracted from the temporal database.

Example. *How long has the Roseanne show run?*

```
SELECT SNAPSHOT ShowName,
    CAST(VALID(N) TO INTERVAL DAY)
FROM NBCShows(ShowName) AS N
WHERE N.ShowName = 'Roseanne'
```

This query has three interesting constructs. The `SNAPSHOT` reserved word indicates that even though the query is defined on a time-varying table, the result is to be a conventional table. The `(ShowName)` found in the from clause requests that the underlying table, `NBCShows`, be *coalesced* on the `ShowName` attribute. Coalescing is similar to conventional projec-

that the viewers will be maximally attentive during that time, though there is also the probability that the commercial will not run.

```
SELECT SUM(NBCShows.Cost), SUM(ABCShows.Cost), SUM(CBSShows.Cost)
FROM NBC_FB_Insertion AS N, NBCShows, ABC_FB_Insertion AS A, ABCShows,
        CBS_FB_Insertion AS C, CBSShows
WHERE N.GameName = NBCShows.ShowName AND A.GameName = ABCShows.ShowName AND
        C.GameName = CBSShows.ShowName AND
    N.InsertionLength = NBCShows.InsertionLength AND
        A.InsertionLength = ABCShows.InsertionLength AND
        C.InsertionLength = CBSShows.InsertionLength
GROUP BY VALID(N) USING MONTH
```

Figure 3: Monthly Distribution Over Vehicles

This query uses tables for the networks ABC and CBS that are analogous to those introduced earlier for NBC. Tables `ABC_FB_Insertion` and `CBS_FB_Insertion` are to the granularity of an `HOUR`, the `ABCShows` table is to the granularity of `ABCSeason`, and the `CBSShows` is to the granularity of `CBSSeason`. The associated calendars determine which days these seasons overlap, to determine which cost to use in computing the cost of a particular football insertion; the costs are then aggregated over each month, as specified by the group by clause, with the `SQL92` calendar determining in which month each day occurs. The default transaction-time selection is to retrieve the known information, that is, the current media plan, rather than some previous version. □

5 Support for Indeterminacy

Often the marketing department will hear about the possible changes to the strategies of competitors, such as the impending introduction of a new light beer. Analyses then need to take this information into account.

The following table records the projected budgets of competing brands.

```
CREATE TABLE TVBudget
    (Company CHARACTER ( 30 ) NOT NULL,
    Brand CHARACTER ( 30 ),
    Network CHARACTER ( 4 ),
    AggregateBudget INTEGER)
VALID STATE NONSTANDARD INDETERMINATE
    YEAR(2) TO MONTH ;
```

Since the marketing department cannot be sure when a competitor's marketing campaign will commence, the `TVBudget` state table is specified as *indeterminate*. Valid-time indeterminacy is "don't know when" indeterminacy. The table thus is specified as an `INDETERMINATE` table; the `NONSTANDARD` reserved word specifies that various distributions other than the standard one may be needed.

Let's assume hypothetically that the intelligence from the field indicates that Jamaican Brewing is planning a big market introduction of a light beer, with a budget of $3 million, but it is not clear when it will start.

```
INSERT INTO TVBudget
VALUES ('Jamaican Brewing',
    'Lowinbrew',
    'NBC',
    3000000)
VALID '1995-01 ~ 1995-04 - 1995-06'
    WITH DISTRIBUTION PROBABLY_EARLY ;
```

Literals express an indeterminate period by indicating the beginning and ending instants, either of which may be indeterminate. The literal specified for the timestamp of the inserted row specifies that the campaign may start as early as January, 1995, or as late as April 1995. Once the campaign begins, it will continue through June, 1995. The `PROBABLY_EARLY` distribution is also specified; this distribution was previously defined by the data base administrator. In this distribution, earlier instants are more likely than later ones.

The underlying granularity is a `MONTH`; the range is 100 years. Such period timestamps, which can be stored in three 32-bit words, contain the range of dates and a probability distribution identifier for both delimiting events.

Queries on indeterminate periods take the indeterminate portion and the probability distribution into account when evaluating predicates. Such a capability is critical when making decisions based on incomplete data.

Example. *Which football insertions on NBC will likely come after the Lowinbrew product introduction?*

```
SELECT SUM(N.Cost), SUM(N2.Cost)
VALID VALID(NI)
FROM NBC_FB_Insertion AS NI NI2, NBCShows AS N N2
WHERE NI.GameName = N.ShowName AND VALID(NI) OVERLAPS VALID(N) AND
    NI2.GameName = NS2.ShowName AND VALID(NI2) OVERLAPS VALID(N2) AND
    TRANSACTION(NI2) OVERLAPS DATE 'now - 14 days' AND
    TRANSACTION(N2) OVERLAPS DATE 'now - 14 days'
GROUP BY VALID(NI) USING MONTH
```

Figure 4: Comparison of Monthly Budgets Between Media Plan Versions

9 Summary

In this tutorial, we have touched on the following facilities and constructs available in the consensus temporal query language TSQL2.

- *Periods* are anchored durations of time. This is a new data type, augmenting SQL's datetimes and intervals.
- *Event tables* are timestamped with sets of instants.
- *State tables* are timestamped with *temporal elements*, which are sets of periods.
- *Bitemporal tables* are timestamped with both valid time and transaction time.
- A *granularity* is a partitioning of the time line.
- *Calendars* provide a collection of granularities, and participate in parsing and output of timestamp literals.
- *Timestamp formats* are user-specified, allowing different applications to control how timestamps are displayed.
- *Valid-time selection* enables rows to be selected by means of predicates on their timestamps, within the where clause.
- *Valid-time projection* specifies the period of validity of a derived table, via the valid clause.
- *Coalescing* merges the timestamps of *value-equivalent* rows, and is specified by listing column names in the from clause.
- *Partitioning* extracts maximal period(s) from a temporal element timestamp for a row, and is specified via the `PERIOD` reserved word in the from clause.
- *Coupled correlation names* permits a further coalescing on additional columns, while ensuring that the rows associated with the two correlation names agree on the values of the original coalescing columns.
- Conventional (non-time-varying) tables can be derived from time-varying tables by specifying snapshot in the select clause. Conventional tables can also participate along with time-varying tables in a select statement.
- *Time-varying aggregates* can be computed. Grouping can be over columns or over row timestamps.
- *Temporal indeterminacy* is supported in the data model, via instant timestamps that encode the first and last possible occurrence times, as well as a probability distribution, and in the query language, via a with plausibility phrase in the where clause.
- *Transaction-time selection* permits specification of previous versions.
- *Now-relative times* are bound during query evaluation.
- *Schema versioning* allows tables timestamped with transaction time to be accessed and modified through previous schemas, thereby supporting legacy applications.
- Tables timestamped with transaction time may be *vacuumed* to remove old versions.

10 Acknowledgements

The domain expertise of Merrie L. Brucks was critical in developing this example, and her assistance is greatly appreciated.