



KONFERENCA
PORTOROŽ, 15. DO 17. MAJ 2017

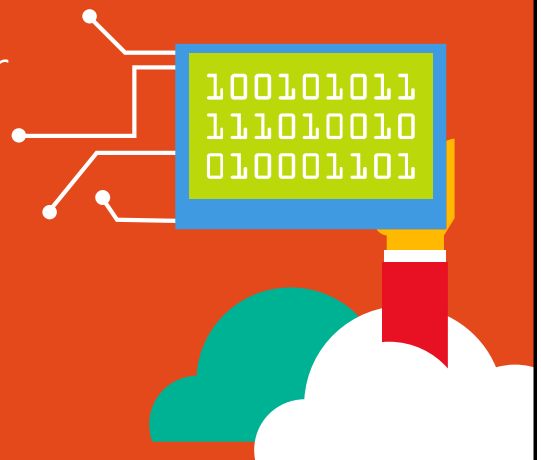


KONFERENCA



Temporal Data in SQL Server

Dejan Sarka



TEHNOLOGIJA

Introduction

- Dejan Sarka
 - dsarka@solidq.com, dsarka@siol.net, @DejanSarka
 - MCT, Data Platform MVP
 - 30 years of data modeling, data mining and data quality
- 14 books, writing more
- 15+ courses



Reach us with #ntk17



Agenda

- Background and references
- Temporal databases
- Allen's interval operators
- Optimizing problems
- SQL Server 2016 temporal tables
- Temporal tables as DW dimensions



Reach us with #ntk17



Theoretical Background

- James F. Allen: [Interval Algebra](http://goo.gl/X7GrQb), "Maintaining knowledge about temporal intervals". *Communications of the ACM* 26(11) pp.832-843, Nov. 1983 (<http://goo.gl/X7GrQb>)
- Chris J. Date, Hugh Darwen, Nikos Lorentzos: [Temporal Data & Relational Databases](http://goo.gl/YsgBxk), Morgan Kaufmann, Nov. 2002 (<http://goo.gl/YsgBxk>)
- Hans-Peter Kriegel, Marco Pötke, Thomas Seidl: [Managing Intervals Efficiently in Object-Relational Databases](http://goo.gl/4mkDEU), Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 2000 (<http://goo.gl/4mkDEU>)
- Laurent Martin: [A Static Relational Interval Tree](http://goo.gl/qjgMm), whitepaper, Nov. 2011 (<http://goo.gl/qjgMm>) and [Advanced Interval Queries with the Static Relational Interval Tree](http://goo.gl/f8X3J4), whitepaper, Jan. 2013 (<http://goo.gl/f8X3J4>)



Reach us with #ntk17



SQL Server Solutions

- Dejan Sarka: [Inside SQL Server 2008: Transact-SQL Programming](http://goo.gl/5TShFO), chapter 12, Temporal Support in the Relational Model, Microsoft Press, Oct. 2009 (<http://goo.gl/5TShFO>)
- Itzik Ben-Gan: [Interval Queries in SQL Server](http://goo.gl/jE3scy), SQL Server magazine article, Jun. 2013 (<http://goo.gl/jE3scy>)
- Davide Mauri: [Using Spatial Data for Interval Queries](http://goo.gl/CEuiFr), blog, Jul. 2013 (<http://goo.gl/CEuiFr>)
- Dejan Sarka: Various methods for optimizing temporal queries, Pluralsight course Working with Temporal Data in SQL Server (<http://goo.gl/XdeRbl>)



Reach us with #ntk17



Temporal Databases

- Many relational databases show current state only
 - Constraints are valid in present, past and future time
 - E.g., supplier A is under a contract
- Temporal databases hold *time-stamped* propositions
 - *Since* predicate means ever since and not immediately before
 - E.g., supplier A is under a contract since a specific time point A
 - *During* (or *from ... to*) predicate means throughout and not immediately before or immediately after
 - E.g., supplier A is under a contract from time point A to time point B
- *Timeline* consists of discrete, indivisible time quanta
 - *Time quantum* is the smallest represented unit
 - Can represent time quanta with integers



Reach us with #ntk17



Semi-Temporal Problems

- Example: suppliers, supplied parts

S#	Since
S1	d04
S2	d07

S#	P#	Since
S1	P1	d04
S1	P2	d05
S2	P1	d08
S2	P2	d09

- No problem with PKs and FKs
- Need additional constraint: no supplier can supply a part before the supplier is under contract



Reach us with #ntk17



Full-Temporal Problems (1)

S#	From	To
S1	d04	d10
S2	d02	d04
S2	d07	d10

S#	P#	From	To
S1	P1	d04	d10
S1	P2	d05	d09
S2	P1	d08	d10
S2	P2	d09	d10
S2	P2	d03	d03

- Two candidate keys in each table (BCNF)
- *To* should not be less than *From*
- Cannot end contract on one day and begin on the next day
 - No abutting



Reach us with #ntk17



Full-Temporal Problems (2)

- No supplier can be under two distinct contracts on the same day
 - No overlapping
- No supplies if not under contract
 - $SP(S\#, From)$ is not a FK!
 - Inclusion dependency:

$$SP(S\#, From, To) \subseteq S(S\#, From, To)$$



Reach us with #ntk17



Valid and Transaction Times

- Bi-temporal databases support valid and transaction times
 - Valid times are human times
 - Transaction times are database times
 - Valid times are updatable
 - Transaction times are not updatable
- In model, we should not need transaction times
 - Should be implemented in a RDBMS
 - Support for valid times would also be appreciated



Reach us with #ntk17



Allen's Interval Boolean Operators

Name	Notation	Definition
Equals	$(i_1 = i_2)$	$(b_1 = b_2) \text{ AND } (e_1 = e_2)$
Before	$(i_1 \text{ before } i_2)$	$(e_1 < b_2)$
After	$(i_1 \text{ after } i_2)$	$(i_2 \text{ before } i_1)$
Includes	$(i_1 \supseteq i_2)$	$(b_1 \leq b_2) \text{ AND } (e_1 \geq e_2)$
Properly includes	$(i_1 \supset i_2)$	$(i_1 \supseteq i_2) \text{ AND } (i_1 \neq i_2)$
Meets	$(i_1 \text{ meets } i_2)$	$(b_2 = e_1 + 1) \text{ OR } (b_1 = e_2 + 1)$
Overlaps	$(i_1 \text{ overlaps } i_2)$	$(b_1 \leq e_2) \text{ AND } (b_2 \leq e_1)$
Merges	$(i_1 \text{ merges } i_2)$	$(i_1 \text{ overlaps } i_2) \text{ OR } (i_1 \text{ meets } i_2)$
Begins	$(i_1 \text{ begins } i_2)$	$(b_1 = b_2) \text{ AND } (e_1 \leq e_2)$
Ends	$(i_1 \text{ ends } i_2)$	$(e_1 = e_2) \text{ AND } (b_1 \geq b_2)$



Reach us with #ntk17



Allen's Interval Algebra Operators

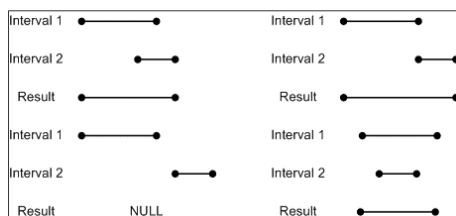
Name	Notation	Definition
Union	$(i_1 \text{ union } i_2)$	$(\text{Min}(b_1, b_2) : \text{Max}(e_1, e_2))$, when $(i_1 \text{ merges } i_2)$; NULL otherwise
Intersect	$(i_1 \text{ intersect } i_2)$	$(\text{Max}(b_1, b_2) : \text{Min}(e_1, e_2))$, when $(i_1 \text{ overlaps } i_2)$; NULL otherwise
Minus	$(i_1 \text{ minus } i_2)$	$(b_1 : \text{Min}(b_2 - 1, e_1))$, when $(b_1 < b_2)$ AND $(e_1 \leq e_2)$; $(\text{Max}(e_2 + 1, b_1) : e_1)$, when $(b_1 \geq b_2)$ AND $(e_1 > e_2)$; NULL otherwise



Reach us with #ntk17

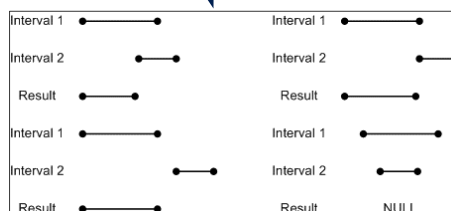


Allen's Interval Algebra Operators

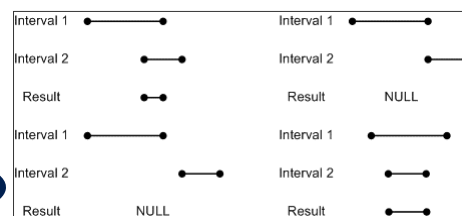


Union

Minus



Intersect



Reach us with #ntk17



Classical T-SQL Solution

- Intervals represented with two columns – begin and end [b, e]
 - Representation with integers
- Find all intervals [b, e] that overlap with a given interval [@b, @e]
- Classic predicate: WHERE b <= @e AND @b <= e

$$(i1 \xleftrightarrow{\text{overlaps}} i2) \equiv (b1 \leq e2 \wedge b2 \leq e1)$$



Reach us with #ntk17



Optimizing Problem

- Optimal indexes exist
 - IDX1 – key b, included e
 - IDX2 – key e, included b
- Problem: two range predicates (WHERE b <= @e AND @b <= e)
 - Only one index is used
 - E.g., use IDX1, seek for b, and then use e as the residual predicate to filter while scanning the rows after the seek
 - Such a seek is very efficient if @e is high
 - Otherwise, IDX2 could be used
 - Indexes are efficient only if they eliminate most of the rows before the scan of the residual rows – if the [@b, @e] is close to the beginning or to the end of the timeline
 - Inefficient around the middle of the timeline
 - Must prevent parameter sniffing



Reach us with #ntk17



Enhanced T-SQL Solution (1)

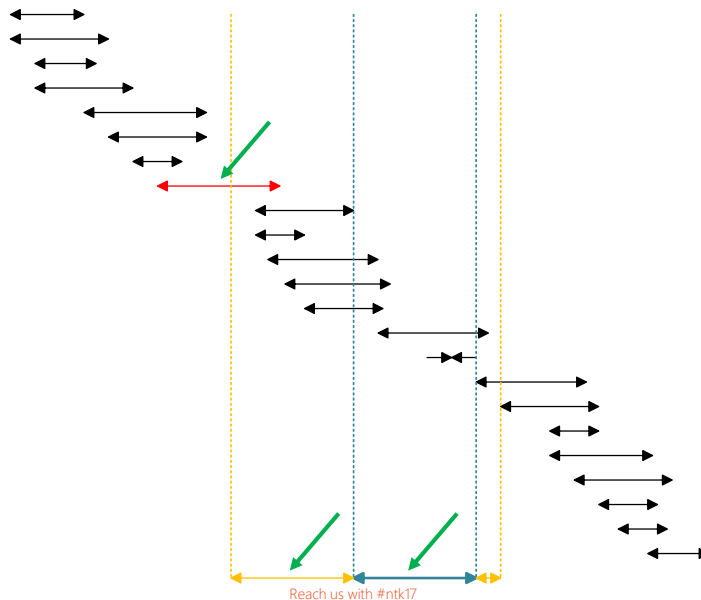
- A simple idea: try to use a single index to eliminate non-matching intervals from both sides
- Eliminating using the [lower] index
 - Eliminating intervals from the right side: just eliminate all intervals where the beginning is at least one unit bigger (more to the right) of the end of the given interval
 - In order to use the same index for eliminating from the left, you need to use the beginning of the intervals in the table in the *WHERE* clause of the query
 - Go to the left side away from the beginning of the given interval at least for the length of the longest interval in the table
 - The intervals that begin before the beginning of the given interval minus the length of the longest interval cannot overlap with the given interval



Reach us with #ntk17



Enhanced T-SQL Solution (2)



Reach us with #ntk17



Enhanced T-SQL Solution (3)

- Can get extremely efficient queries
- The solution is simple
- However, the performance drops substantially as soon as you have one very long interval in your table
- Therefore, the solution efficiency depends on the interval length distribution
 - The more uniform distribution, the more efficient solution



Reach us with #ntk17



SQL Server Temporal Tables

- System-versioned tables store a full history of data changes
 - A validity period, is stored in two datetime2 columns – SysStartTime and SysEndTime
- All versioning is automatic, no change in the code needed
 - Can be added to existing tables
- History is stored in an associated table
 - Historical table can be named, or take system name



Reach us with #ntk17



Considerations and Limitations (1)

- The SysStartTime and SysEndTime columns must use the datetime2 data type
- The table must have a primary key; the historical table cannot use constraints
- To name the history table, specify the schema and table names
- History table compression defaults to PAGE
- The history table must reside in the same database as the current table



Reach us with #ntk17



Considerations and Limitations (2)

- System-versioned tables are not compatible with FILETABLE or FILESTREAM
- Columns with a BLOB data type, such as varchar(max) or image, can result in high storage requirements
- INSERT and UPDATE statements cannot reference the SysStartTime or SysEndTime columns
- You cannot directly modify data in the history table
- You cannot truncate a system-versioned table
- Merge replication is not supported



Reach us with #ntk17



Creating a Temporal Table

```
CREATE TABLE dbo.Employee
(
    EmployeeID int NOT NULL PRIMARY KEY CLUSTERED,
    ManagerID int NULL,
    FirstName varchar(50) NOT NULL,
    LastName varchar(50) NOT NULL,
    SysStartTime datetime2
        GENERATED ALWAYS AS ROW START NOT NULL,
    SysEndTime datetime2
        GENERATED ALWAYS AS ROW END NOT NULL,
    PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime)
)
WITH (SYSTEM_VERSIONING =
    ON (HISTORY_TABLE = dbo.EmployeeHistory));
```



Reach us with #ntk17



Querying Temporal Tables

- System-versioned tables can be queried using the new FOR SYSTEM_TIME clause in the FROM part of a query
- Use one of the following four sub-clauses:
 - AS OF <date_time>
 - FROM <start_date_time> TO <end_date_time>
 - BETWEEN <start_date_time> AND <end_date_time>
 - CONTAINED IN (<start_date_time>, <end_date_time>)
 - ALL to return everything



Reach us with #ntk17



Temporal Tables as Dimensions

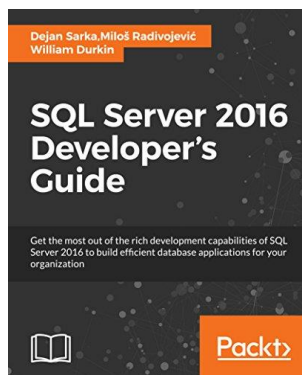
- System-versioned tables can be out of the box solution for Type 2 SCD problem
 - Can simplify the ETL process
- However, SSAS does not support temporal tables yet
 - You are limited to query the data warehouse directly, with T-SQL queries
- System versioning works on row level
 - Might need column-level versioning



Reach us with #ntk17



Q & A



- Only for attendees: 40% discount on e-book!
 - Discount Code: **EBSQLS40**
 - Start date: May 1st 2017
Expiry date: May 30th 2017
 - Create a login on the Packt site www.packtpub.com and add the book to the cart
 - Click "View Cart"
 - "Do you have a promo code?" field enter (code provided above)
 - Click the "Apply" button to apply the discount

- Last SQL Saturday of the year: December 9th, 2017



Reach us with #ntk17





© Copyright Microsoft Corporation. All rights reserved.