BitemporalData.com

An Overview of Temporal Features in SQL:2011

Craig Baumunk, January 4, 2012

Agenda

- SQL Standards Process Overview
- Temporal features in standard SQL history
- Temporal data overview
- Things to know about temporal features in SQL:2011
- Application time period tables
- System versioned tables
- Application time period tables with system versioning (bitemporal tables)
- Next Steps

SQL Standards Process Overview

- INCITS DM32.2 (formerly X3H2) is responsible for SQL standard in US
  - International Committee for Information Technology Standards) is an ANSI (American National Standards Institute) accredited standards producing organization
- ISO/IEC JTC 1/SC 32 Data Management and Interchange/WG 3 committee is responsible for SQL Standard Internationally
- Much of the new capabilities in the SQL standard have originated in the US
  - Approval by DM32.2 before submission to WG3
  - Published by ISO then adopted as US standards by INCITS
- Typically 3 to 5 year cycle
- SQL is a multi-part standard which currently has 9 parts
  - The highest part number is currently 14 (parts 5, 6, 7, 8 &12) were terminated
  - Part 2: SQL/Foundation is the SQL language specification (biggest/most important part)
- Documentation of DM32.2 works in progress are not publicly available
- Anyone can join DM32 as an observer for $1,200 and get documentation of works in progress and what is approved but not yet published etc
- 7 Versions of Standard SQL
  - 86 (SQL-87), 89, 92(SQL2), SQL:1999 (SQL3), SQL:2003, SQL:2008, SQL:2011

* Disclaimer: I am not and never been a member of X3H2 or INCITS DM32.2 personally

Temporal Features First Attempt 1995 - 2001

- X3H2 (now DM32.2) and WG 3 both approved work on a new part of SQL standard called SQL/Temporal in 1995
- US made the first proposal on adding the new SQL extensions, largely based on the pioneering work of Prof. Rick Snodgrass of Univ. of Arizona
- The US proposal was based on TSQL2, an extension of SQL-92, put together by a team headed by Prof. Snodgrass
- The US proposal proved to be controversial at ISO. Some of the ISO members felt there were some serious problems with the US proposal
- The UK brought in a competing proposal based on the work of Prof. Nikos Lorentzos of Univ. Athens, Greece
- The US disagreed with the ISO comments on the US proposal. And the US also did not see the need for the UK proposal
- Because of the controversy, both ANSI and ISO decided to defer further work on SQL/Temporal until SQL:99 was published
- After the publication of SQL:99, neither US nor UK brought in any new proposals to resolve the differences
- Because of inactivity, both ANSI and ISO decided to cancel SQL/Temporal part in 2001

Temporal Features Second Attempt 2008 - 2011

- A second attempt at adding temporal features to the SQL standard was made in 2008. It started with the acceptance of a proposal on "system-versioned tables" by both INCITS DM32.2 and ISO/IEC JTC1 SC32 WG3. Rather than resurrecting SQL/Temporal, this proposal added the temporal extensions to SQL/Foundation.

- Another temporal feature was added in 2010 in the form of "application-time period tables".

- Both system-versioned tables and application-time period tables are now part of the new version of the SQL standard (SQL:2011) which was approved and published in 2011.

- The temporal features in SQL:2011 are largely inspired by the earlier proposals considered during the first attempt but with a substantially different syntax.

Temporal Data

- ❑ **Temporal data is data which changes over time**
  - A company's credit rating changes over time
    - Referred to as valid time dimension, real world perspective or business perspective
    - SQL:2011 calls this application time
  - The value we have in a database for a company's credit rating changes over time
    - This may be different than application time due to timing differences
    - This may have nothing to do with the application time perspective (for example corrections)
    - Referred to as transaction time dimension or database perspective
    - SQL:2011 calls this system time (or system versioning)

- ❑ **Most data is temporal**
  - Most common dimensions are application time and system time (focus of temporal features of SQL:2011)

- ❑ **It is helpful to think of non-temporal data as a trivial case of temporal data**
  - Non-temporal data would be data which does not change (in real life, in a system, <u>or we do not store a history of changes in a database</u>)

- ❑ **Bitemporal data is data which changes over 2 dimensions of time independently**
  - SQL:2011 calls this application time with system versioning

4 Types of tables

BitemporalData.com

	No Application Time History	Application Time History
No System Time History	Non-temporal	Application time period table
System Time History	System versioned table	System versioned application time period table

❑ **Non-temporal tables**

- Conventional tables (without date/timestamp in PK)
- Latest information, without any history

❑ **Application time period tables**

- History of how data changed from an application time perspective as we know it now

❑ **System versioned tables**

- History of how current data from a application time perspective changed in the system (database)

❑ **System versioned application time period tables**

- History, of how history from an application time perspective, changed in the system (database)

Things to know about temporal features in SQL:2011

❑ **Row based versioning**

- In contrast to column based versioning
- But can be use to implement column based versioning

❑ **State based storage**

- 2 timestamps for application time periods
- 2 timestamps for system time periods
- 4 timestamps for system versioned application time period tables

❑ **State based inputs**

- User provides application period start AND application period end
- Users do NOT provide system period start or end
    - Insert, update, and delete events are converted to states (till high datetime)

❑ **Time periods use the Closed – Open convention**

- AKA inclusive – exclusive
- Supports unambiguous comparisons with different precisions

❑ **The strengths of the temporal features in SQL:2011 are:**

- Ease of migration (not dependent on period data type)
- Fits well with SQL semantics (no statement modifiers)
- Lots of user control (can update application start/end directly)
- Compatibility (existing queries work against system versioned tables)

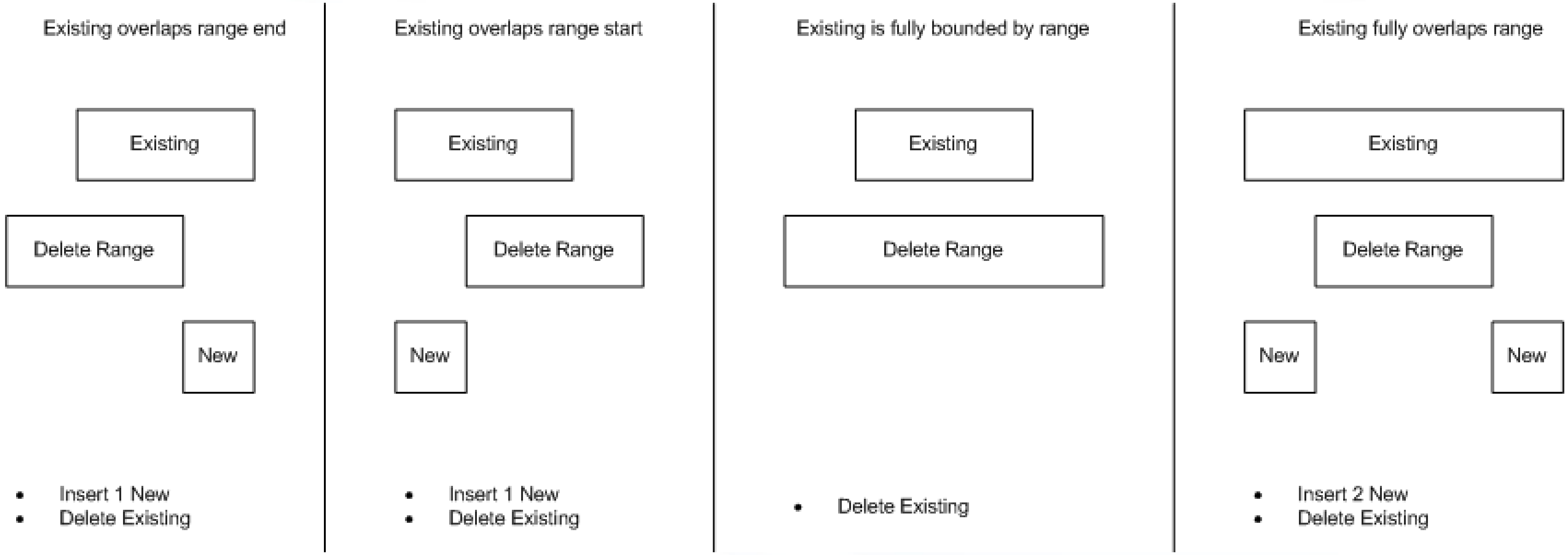
Primary Keys for 4 Types of Tables

- ❑ **Non-temporal storage**
  - No date/timestamp in PK
- ❑ **Application time period table**
  - 2 timestamps (Application Period Start & Application Period End) is not enough
  - Application time periods can not overlap for the same object
- ❑ **System versioned table**
  - System time periods should not overlap for the same object (but this is not part of the PK)
  - 2 rows for the same object can not have the same system start (constraint)
    - Transaction time is defined by implementation (high level of precision)
    - Transaction times should be sequential
- ❑ **System versioned application time period table**
  - Application time periods can overlap for the same object
  - System time periods can overlap for the same object
  - BOTH application time period AND system time period should NOT overlap for the same object

Application time period tables

- Table definition
  - Define columns for application period start and end (need to be a date/timestamp data types and not null)
  - New PERIOD clause specifies what columns implement an application time period and creates an implicit constraint that application time period start < application time period end (pick your period name)
  - New WITHOUT OVERLAPS keywords in PRIMARY KEY clause ensures that application time periods do not overlap
  - FOREIGN KEY clause extended to include referencing of period in parent table
- Inserting
  - Normal syntax
    - Comply with constraints (start and end not null, start < end, no overlaps)
- Deletes
  - Delete rows with normal syntax
    - Constrain on application time period start and end as desired or not !
  - Delete using FOR PORTION OF application time period FROM DATE XXXX/XX/XX TO DATE XXXX/XX/XX clause results in automatic “row splitting”
- Updates
  - Update rows with normal syntax (including application time period start and end columns)
    - Comply with constraints (start and end not null, start < end, no overlaps)
    - Constrain on application time period start and end as desired or not !
  - Update using FOR PORTION OF application time period FROM DATE XXXX/XX/XX TO DATE XXXX/XX/XX clause results in automatic “row splitting”
    - Can’t update application period start or end using this option
- Select
  - Normal Syntax (application time period start and end can be used since they are explicit columns)

Application time period tables – Delete row splitting
- 4 types of existing time slices -



Existing overlaps range end

Existing

Delete Range

New

- Insert 1 New
- Delete Existing

Existing overlaps range start

Existing

Delete Range

New

- Insert 1 New
- Delete Existing

Existing is fully bounded by range

Existing

Delete Range

- Delete Existing

Existing fully overlaps range

Existing

Delete Range

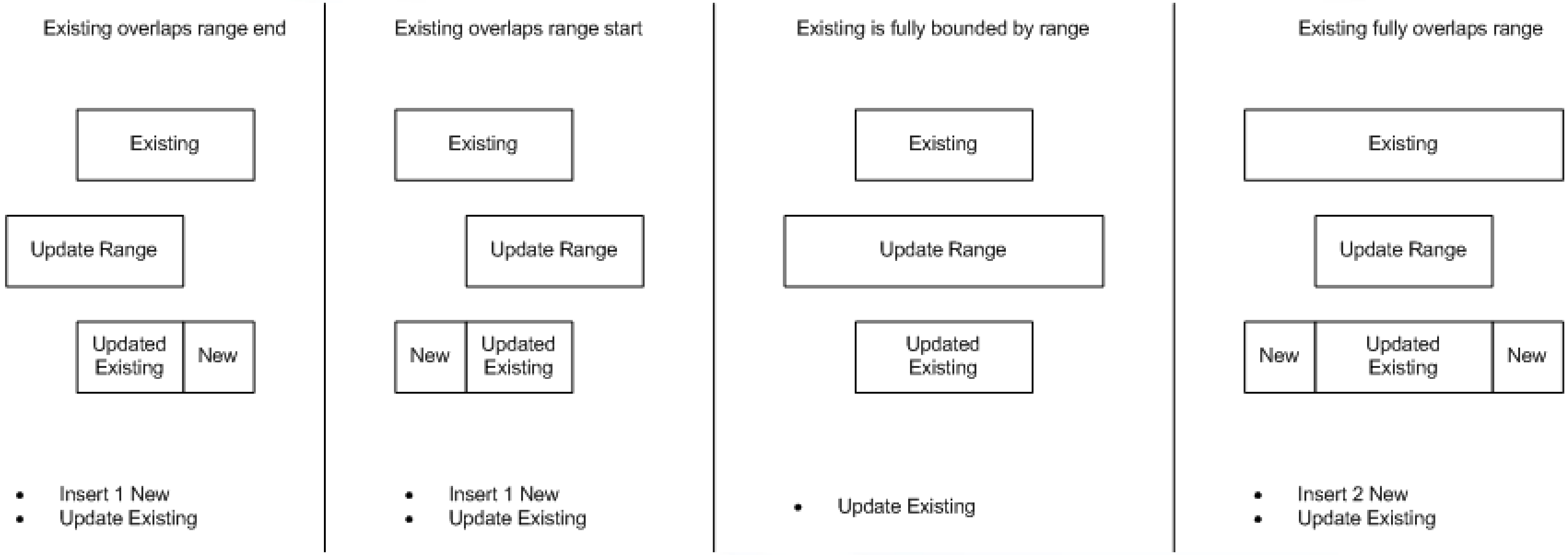
New

New

- Insert 2 New
- Delete Existing

- There can be multiple types of existing time slices for a delete range
- No error if delete over a range where no values

Application time period tables – Update row splitting - 4 types of existing time slices -

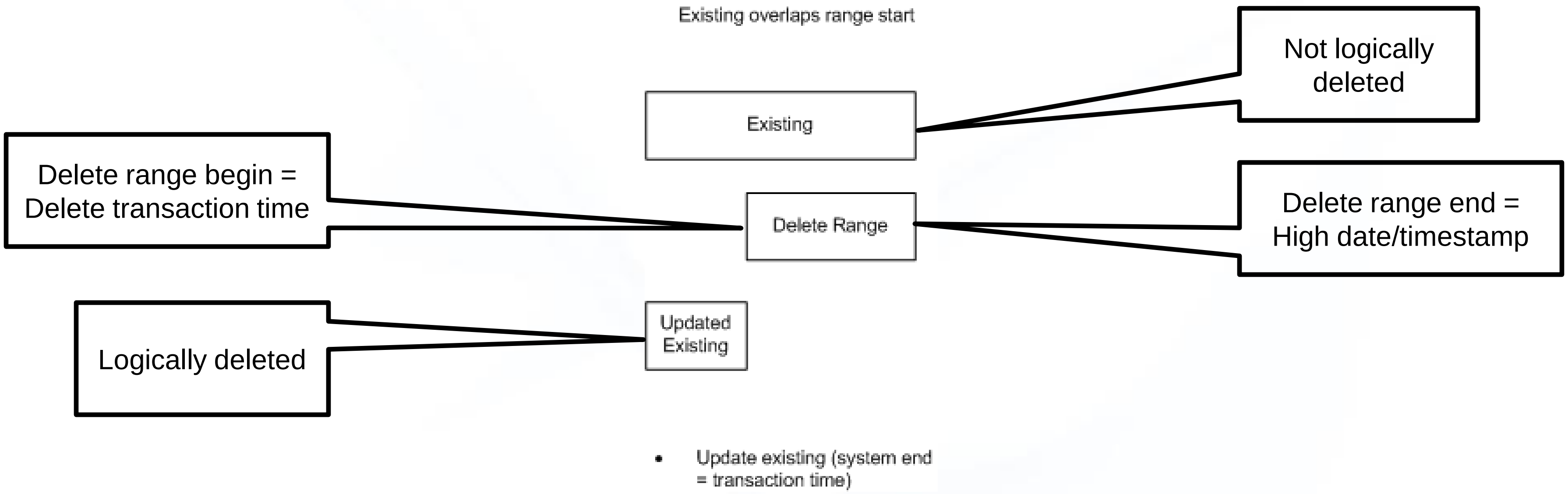


- There can be multiple types of existing time slices for a update range
- No error if update over a range where no values
- An update can cause multiple contiguous time slices to have the same non PK data values (unpacked)

System versioned tables

- Table definition
  - Define columns for system time period start and end (need to be date/timestamp data type and not null) with new GENERATE ALWAYS AS ROW BEGIN/END clause
  - New PERIOD clause with SYSTEM_TIME period name specifies what columns are in the system time period (system time period start < system time period end does not need to be enforced)
  - New WITH SYSTEM VERSIONING clause implicitly adds system time period start to primary key
  - Constraints only apply to rows which are not logically deleted
  - Referential Integrity is not impacted
- Inserting
  - Normal syntax but never specify system time period start or end columns as they are generated
  - New time slices of data have system time period start = insert transaction time and system period end = high date/time
- Deletes
  - Normal syntax but a history of what data was deleted and when it was deleted is maintained automatically
  - Deleted time slices of data have system period end = delete transaction time (logically deleted)
  - Only rows which are not logically deleted can be deleted
- Updates
  - Normal syntax but a history of what data was updated and when it was updated is maintained automatically
  - Updated time slices of data have system period end = update transaction time (logically deleted)
  - Only rows which are not logically deleted can be updated
- Select
  - Normal syntax works as if non-temporal table (do not query logically deleted rows)
  - New as-of, between, and from clauses are available to query all rows in the table
    - Including logically deleted rows

System versioned tables – Delete history
- 1 type of existing time slice -



Existing overlaps range start

Existing

Not logically deleted

Delete range begin = Delete transaction time

Delete Range

Delete range end = High date/timestamp

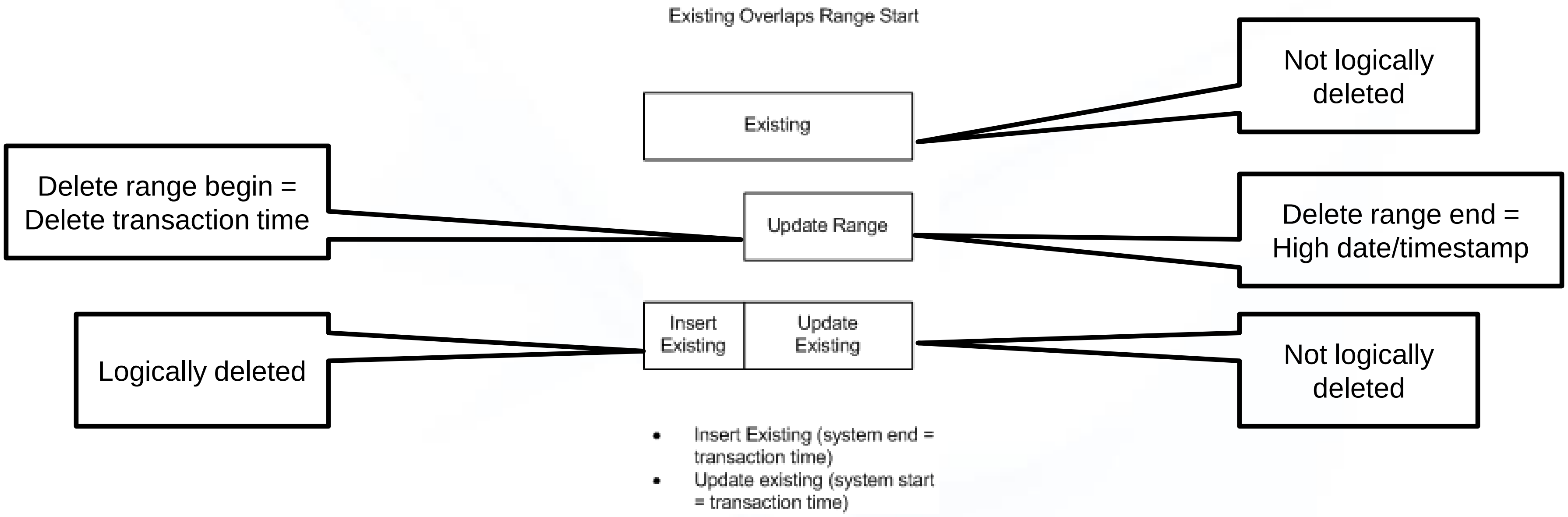
Updated Existing

Logically deleted

- Update existing (system end = transaction time)

- There is only 1 type of existing time slice for a delete range
- There can only be 1 time slice of interest for an object (that is not logically deleted)

System versioned tables – Update History
- 1 type of existing time slices -



Existing Overlaps Range Start

Existing

Update Range

Insert Existing

Update Existing

- Insert Existing (system end = transaction time)
- Update existing (system start = transaction time)

Not logically deleted

Delete range begin = Delete transaction time

Delete range end = High date/timestamp

Logically deleted

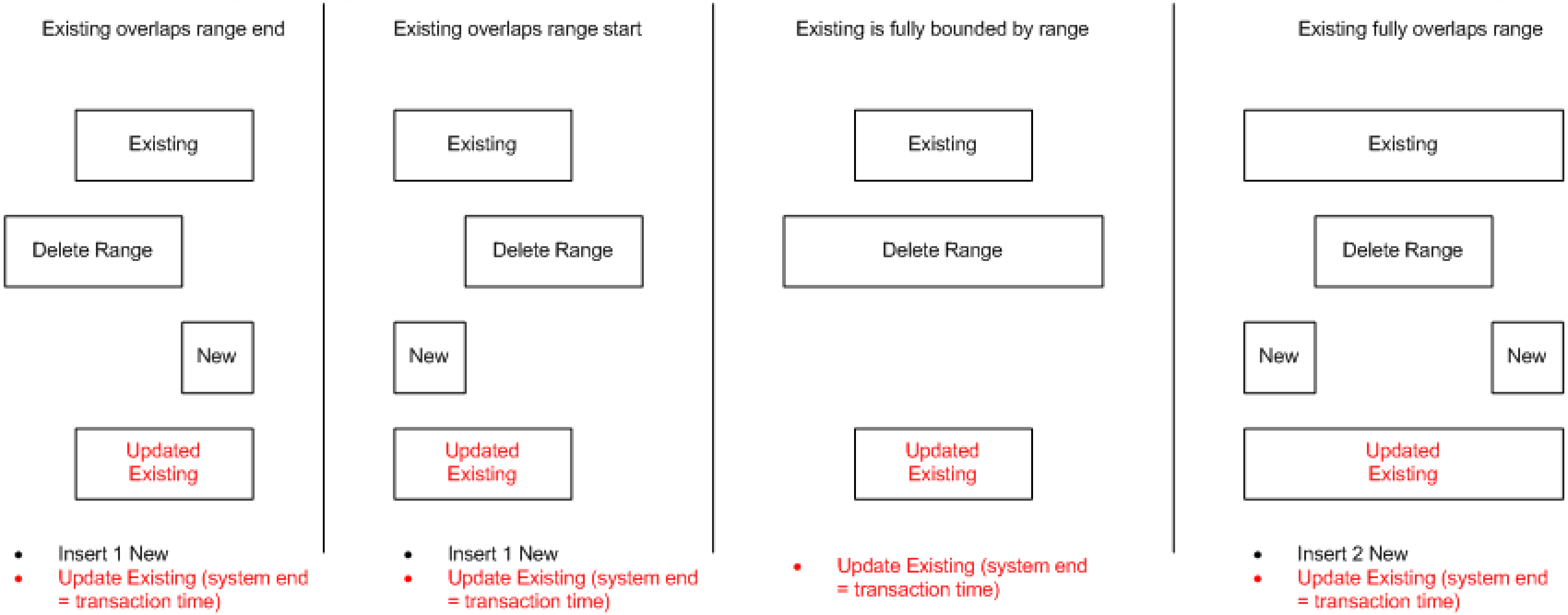
Not logically deleted

- There can only be 1 type of existing time slice for an update range
- There can only be 1 time slice of interest for an object (that is not logically deleted)

System versioned application time period tables

- Table definition
  - Define columns for application time period start and end (need to be date/timestamp data types and not null)
  - Define columns for system time period start and end (need to be date/timestamp data types and not null) with new GENERATE ALWAYS AS ROW BEGIN/END clause
  - New PERIOD clause to specifies what columns are in the application time period
  - New PERIOD clause with SYSTEM_TIME keyword specifies what columns are in system time period
  - New WITHOUT OVERLAPS keyword in PRIMARY KEY clause ensures that application time periods do not overlap
  - New WITH SYSTEM VERSIONING clause implicitly adds system time period start to primary key
  - Constraints only apply to rows which are not logically deleted
  - FOREIGN KEY clause extended to include referencing of period in parent table
- Inserting
  - Same as application time period tables but never specify system time period start or end columns as they are generated
- Deletes
  - Same as application time period tables but a history of what was deleted and when it was deleted is maintained automatically (rows)
  - Only rows which are not logically deleted can be deleted
- Updates
  - Same as application time period tables but a history of what was updated and when it was updated is maintained automatically (rows)
  - Only rows which are not logically deleted can be updated
- Select
  - Same as application period tables
  - New as-of, between, and from clauses are available to query all rows in the table
    - Including logically deleted rows

Bitemporal tables – Delete splitting & history
- 4 types of existing application time slices -



Existing overlaps range end

Existing

Delete Range

New

Updated Existing

- Insert 1 New
- Update Existing (system end = transaction time)

Existing overlaps range start

Existing

Delete Range

New

Updated Existing

- Insert 1 New
- Update Existing (system end = transaction time)

Existing is fully bounded by range

Existing

Delete Range

Updated Existing

- Update Existing (system end = transaction time)

Existing fully overlaps range

Existing

Delete Range

New

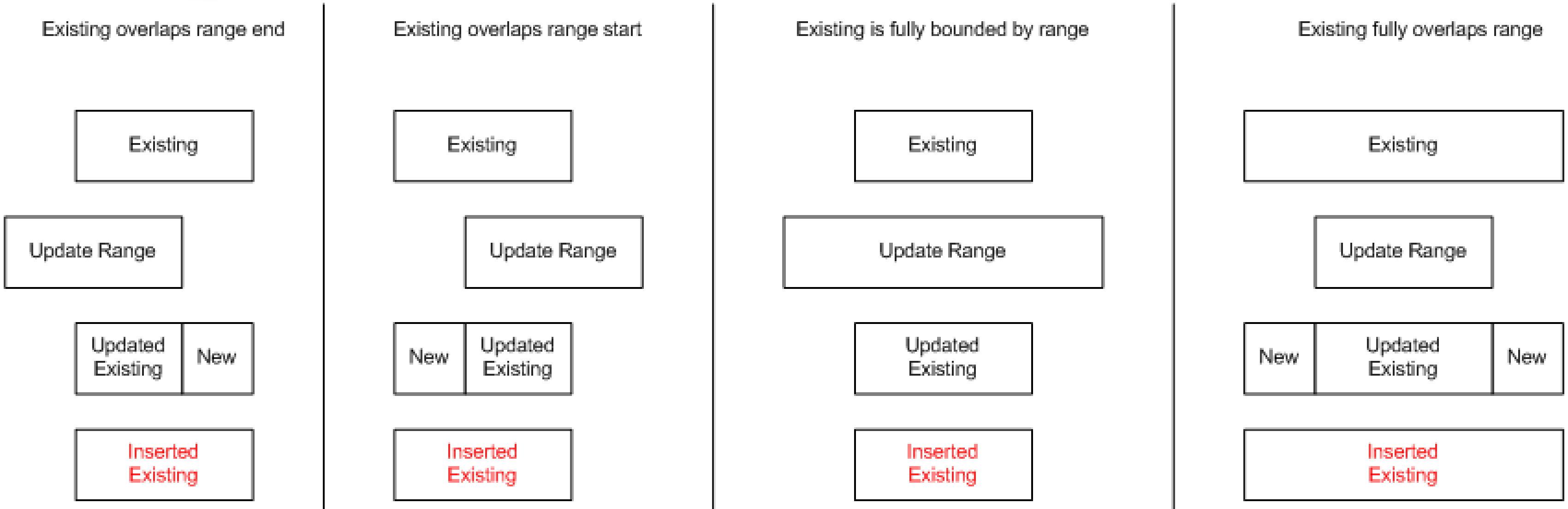
New

Updated Existing

- Insert 2 New
- Update Existing (system end = transaction time)

- Same as application time period table deletes except update existing time slices instead of delete because:
  - Only 1 type of existing row from system versioning perspective
  - Application time period table delete splitting results in inserts and deletes
  - System versioning does not do any splitting for inserts
  - System versioning changes deletes into updates

Bitemporal tables – Update splitting & history
- 4 types of existing application time slices -



Existing overlaps range end

Existing

Update Range

Updated Existing | New

Inserted Existing

- Insert 1 New
- Update existing (system start = transaction time)
- Insert existing (system end = transaction time)

Existing overlaps range start

Existing

Update Range

New | Updated Existing

Inserted Existing

- Insert 1 New
- Update existing (system start = transaction time)
- Insert existing (system end = transaction time)

Existing is fully bounded by range

Existing

Update Range

Updated Existing

Inserted Existing

- Update existing (system start = transaction time)
- Insert existing (system end = transaction time)

Existing fully overlaps range

Existing

Update Range

New | Updated Existing | New

Inserted Existing

- Insert 2 New
- Update existing (system start = transaction time)
- Insert existing (system end = transaction time)

- Same as application time period table updates except additional insert of existing time slice because:
  - Only 1 type of existing row from system versioning perspective
  - Application time period table update row splitting results in inserts and updates
  - System versioning does not do any splitting for inserts
  - System versioning changes updates into an update AND an insert

Summary & Conclusions

- Significant new temporal features have been included in SQL:2011
- Temporal features represent a significant extension to the SQL language that will take time for people to utilize
- Vendors are beginning to adopt these features
- Methodology for how to utilize new temporal features of SQL will probably be a factor in utilization
- Time will tell how much the SQL:2011 temporal extensions are utilized
- Implications of temporal extensions on replication, partitioning, archiving etc are still being sorted

Next Steps

- Actual syntax
- Examples, Examples and more examples
- Advanced topics
  - Temporal predicates
  - Joining different types of temporal tables
  - Referential integrity implications
  - Schema migration implications
- Where are vendors with compliance/adoption
- How vendors CAN comply but also differentiate
- How vendors ARE differentiating

Resources

- ❑ Search YouTube.com for “Case for Bitemporal Data”
  - ❑ http://www.youtube.com/playlist?list=PL4CB3C8161D2804E6
- ❑ TemporalData.com
- ❑ “Temporal Data” LinkedIn group
  - ❑ http://www.linkedin.com/groups?gid=3885228
- ❑ http://stacresearch.com/btd
  - ❑ Functional and performance benchmarks

THANK YOU!

USA

BitemporalData.com LLC
1330 Route 206
Suite 318
Skillman, NJ 08558

Tel: +1.201.332.5000

UK

BitemporalData Ltd
Suite 13493, 2nd floor
145-157 St John Street
London EC1V 4PY

Tel: +44.20.3318.5919