# Point-based Temporal Extension of SQL

**Domain:** The Domain of this article is databases. This paper defines a new approach to Temporal Extension of SQL.

**Problem**: Time stamping tuples with single time instants leads to enormous space requirements because the tuples are repeated for each time instant in which the tuple is true and fulfills a fact. And there was a solution by using sets of time instants which presents periods of validity.

**Sub problem approach:** Tension between the syntax and the intended semantics of the language

- Time references are temporal attributes
- Data model and semantics are point-based; intervals are used as compact description of large sets of instants
- Coalescing in a single dimension system facilities representation of queries, but there are queries can't be represented in one dimension

In order to nullify the limitations in recent proposals relating to temporal extensions of SQL, the point-based references to time approach is taken into account. The main idea in this paper is to define a new language SQL/TP which extends the syntax and semantics of SQL/92 and facilitates both existing and new database management systems to enhance the support for temporal data.

**Interest and motivation approach:**

- A language in which temporal attributes range over single time instants92 (add a single type which presents a linearly ordered universe of individual time instants)
- Define meaningful approach to duplicate semantics and aggregation that is independent of the encoding
- There is no need for repeating rows for presenting a period of time

**Contributions:**

- Data model for temporal database:
- Time domain: discrete countable infinite linearly ordered set without end points.
- Individual elements represent the actual time instants.
- The linear order represents the progression of time.
- Time granularity is implementation-dependent.
- Abstract temporal database: is a set of tables where a particular tuple is related to a large or infinite set of time instants for expressing a period, instead we can use compact encoding of sets of time instants which we chose concrete temporal DB ( interval based encoding).
- Concrete temporal database: the internals are represented by *tmin* and *tmax* which specifies endpoints of intervals.

- The language SQL/TP
- Data definition language: is like standard SQL/92, but the difference is that we defined a new data type '*time'* for declaring temporal attributes and has modifiers to determine how the time instances are stored in tables. There are two ways to specify time:
    - Using points: the time is stored as atomic value for representing a single atomic event that happened in a specific time.

- Using [bounded|unbounded] intervals: represents continuous time instants as duration of a specific event. Bounded and unbounded keys specify ∞ and -∞ as an endpoint for the intervals.

We will define a table indep by using interval-based encoding (we assume time instants presented by integers with a fix granularity):

*Create table indep(name char(20), year time using unbounded intervals)*

- Query language: uses two basic syntactic constructs:
  - Select block: same to standard SQL
  - Set operations: same to standard SQL
  - We will use a special constant pseudo-relation: true(t:time) for the temporal domain; by that we can have the complementation over the temporal domain.
- Semantics: SQL/TP is SQL/92 extended with additional data type 'time'; so we can use the familiar SQL-like semantics, and not affecting the syntax and semantics of queries.

- Query Compilation Technique**:**
  - To verify that the queries built in SQL/92 are efficient
  - Definition of a translation procedure that allows compiling SQL/TP statements to standard SQL/92 statements and translation utilizes the quantifier elimination procedure for linear order to replace reference to individual time instants in the queries with references to interval endpoints.

**Work relies and distinctive points:**

SQL/TP can be:

- Implemented on top of an interval-based representation of temporal database.
- Built to an existing RDBMS, and provide temporal capabilities without changing or modifying the underlying database system.
- Express all representation independent SQL/Temporal queries.

**Perspectives and Conclusion**

The article concludes that the abstract view of point based approach to temporal extensions of SQL has advantages over the common approaches that use interval-based attributes. The idea of a new query language SQL/TP gives rise to several other future perspectives mentioned below:

- Use of complex temporal domains
- Optimization technique for SQL/TP
- Area of updates is always cumbersome for any query language. Management of updates in SQL/TP
- Introduction of specialized indices for SQL/TP