ISO/IEC JTC 1/SC 32 N 1294

Date: 2005-04-22

REPLACES: --

ISO/IEC JTC 1/SC 32

Data Management and Interchange

Secretariat: United States of America (ANSI)
Administered by Farance Inc. on behalf of ANSI

DOCUMENT TYPE	Other Document (Open)
TITLE	Presentation – Proposal foe a New SQL/MM Subproject – SQL/MM Part 7: History
SOURCE	WG 4
PROJECT NUMBER	
STATUS	
REFERENCES	
ACTION ID.	FYI
REQUESTED ACTION	
DUE DATE	
Number of Pages	35
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretary, ISO/IEC JTC 1/SC 32
Farance Inc *, 360 Pelissier Lake Road, Marquette, MI 49855-9678, United States of America
Telephone: +1 906-249-9275; E-mail: MannD@battelle.org
available from the JTC 1/SC 32 WebSite http://jtc1sc32.org/
*Farance Inc. administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

ISO/IEC JTC 1/SC 32/WG 4:TXL-014
April 11, 2005

Proposal of a New SQL/MM Subproject ---SQL/MM Part 7: History

April 2005
Berlin, Germany

Y. Ishii, T. Kajino, Y. Noguchi, M. Inui, T. Ishizaka
K. Suzuki, T. Kotera, K. Yamahira and K. Shibano
Japan

The Problem

- During an update operation, the value of an attribute is overwritten by the new value and the "before" value no longer exists
  - When the attributes of people (e.g. employees, customers, patients, and criminals) or products are expressed in a table, and
  - When the values of the attributes (e.g. position title, address, diagnose, crime committed, unit price) change over time

Our Proposal

- We are proposing SQL/MM Part7: History based on SQL language to enable
  - Creating and maintaining a separate table, which we call “history table”, to store the change history of the base table.
  - Providing methods for time-related queries to the history table.

Approach

- It is possible to accumulate the change history of an existing table without changing the existing business application.
- We realize the function using the framework of SQL/MM.
  - Define the procedure HS_CreateHistory to create and maintain the history table.
  - Define the following two user-defined types.
    - HS_History type
    - HS_TYPE_<TableName> type

Several Examples of the General Time-Related Queries

- Which employee(s) lived in San Francisco on December 31, 1999?
- Which employee(s) belonged to the Engineering Department No. 1 for 7 years consecutively?
- Which customer got a promotion to Gold rank during the period between January 1, 2002 through December 31, 2002?
- Which products has price changes 3 times or more in the last 2 years?
- Which patients became ill with Illness zz within 3 years after undergoing Medical Treatment yy for Illness xx?
- After leaving hospital, how many patients with a certain inspection value xx being below yy are still alive 6 months later?

Structure of History Table

- History table is a table that holds change history for the original base table. The history row is inserted into the history table when Update operation to the specified column(s) of the original base table occurs.
- History table is defined as a table which is typed by HS_TYPE_<TableName> type, and consists of the following columns:
  - All columns in the original base table.
  - column of structured type HS_History, which has attributes of history begin time and history end time.

Original base table (current state table)

Key_Col	**Column_A**	**Column_B**	**Column_C**
K0	A0	B0	C0

History table

Key_Col	**Column_A**	**Column_B**	**Column_C**	**HS_Hist**	
				HS_HistoryBeginTime	**HS_HistoryEndTime**
K0	A0	B0	C0	1997-04-05 10:23:51	NULL

Creation and Maintenance of History Table

- History table is created automatically when the HS_CreateHistory procedure is called.
- When a history table is created, all rows in the original base table are inserted with history begin time set to the CURRENT_TIMESTAMP.
- Insertion of history row to history table is automatically executed by triggers that fire when Update/Insert/Delete operation to the original base table is executed.

Original base table (current state table)

Key_Col	**Column_A**	**Column_B**	**Column_C**
K0	A0	B0	C0

History table

Key_Col	**Column_A**	**Column_B**	**Column_C**	**HS_Hist**	
				HS_HistoryBeginTime	**HS_HistoryEndTime**
K0	A0	B0	C0	1997-04-05 10:23:51	NULL

Automatic Maintenance of History Table

- The following five triggers perform automatic insertion of history row to history table, and maintenance of the consistency of history table.

Event	Process
Insert to original base table	A row in the history table is inserted, and history begin time and history end time are set to CURRENT_TIMESTAMP and NULL respectively.
Update to the specified columns of original base table	History end time of the latest history row in the history row set is updated to CURRENT_TIMESTAMP. A row in the history table is inserted, and history begin time and history end time are set to CURRENT_TIMESTAMP and NULL respectively.
Delete in original base table	History end time of the latest history row in the history row set is updated to CURRENT_TIMESTAMP.
Before Update to HS_Hist column of history table	Forbid updating other than the latest history row in the history row set.
After Update to HS_Hist column of history table	Maintain the consistency of the values of history begin time and history end time in the history table.

More Detailed Explanation of Process for Insert Operation(1)

- When Insert operation to the original base table is executed, a row in the history table is inserted.

Original base table (current state table)

Key_Col	Column_A	Column_B	Column_C
K0	A0	B0	C0
K1	A1	B1	C1

History table

Key_Col	Column_A	Column_B	Column_C	HS_Hist	
				HS_HistoryBeginTime	HS_HistoryEndTime
K0	A0	B0	C0	1997-04-05 10:23:51	NULL
K1	A1	B1	C1	1998-01-05 12:38:33	NULL

More Detailed Explanation of Process for Insert Operation(2)

- When Insert operation to the original base table is executed, a row in the history table is inserted.
- When a user wants to specify the value of history begin time of the latest history row in the history row set clearly, the value of history begin time is updated by invoking HS_HistoryBeginTime method provided in the HS_TYPE_<TableName> type.

Original base table (current state table)

Key_Col	Column_A	Column_B	Column_C
K0	A0	B0	C0
K1	A1	B1	C1

History table

Key_Col	Column_A	Column_B	Column_C	HS_Hist	
				HS_HistoryBeginTime	HS_HistoryEndTime
K0	A0	B0	C0	1997-04-05 10:23:51	NULL
K1	A1	B1	C1	1998-01-01 00:00:00	NULL

More Detailed Explanation of Process for Update Operation

- When Update operation to the original base table is executed, history end time of the latest row in the history row set is updated to CURRENT_TIMESTAMP, and a row in the history table is inserted.
- In the same way as the example of Insert operation, if necessary, the value of history begin time can be updated. (History end time of previous history row is also simultaneously update into the same value by the trigger.)

Original base table (current state table)

Key_Col	Column_A	Column_B	Column_C
K0	A0	B0	C0
K1	A2	B2	C1

History table

Key_Col	Column_A	Column_B	Column_C	HS_Hist	
				HS_HistoryBeginTime	HS_HistoryEndTime
K0	A0	B0	C0	1997-04-05 10:23:51	NULL
K1	A1	B1	C1	1998-01-01 00:00:00	2000-01-01 00:00:00
K1	A2	B2	C1	2000-01-01 00:00:00	NULL

More Detailed Explanation of Process for Delete Operation

- When Delete operation to the original base table is executed, history end time of the latest row in the history row set is updated to CURRENT_TIMESTAMP.
- When a user wants to specify the value of history end time of the latest history row in the history row set clearly, the value of history end time is updated by invoking HS_HistoryEndTime method provided in the HS_TYPE_<TableName> type.

Original base table (current state table)

Key_Col	Column_A	Column_B	Column_C
K0	A0	B0	C0

History table

Key_Col	Column_A	Column_B	Column_C	HS_Hist	
				HS_HistoryBeginTime	HS_HistoryEndTime
K0	A0	B0	C0	1997-04-05 10:23:51	NULL
K1	A1	B1	C1	1998-01-01 00:00:00	2000-01-01 00:00:00
K1	A2	B2	C1	2000-01-01 00:00:00	2003-01-01 00:00:00

How to Query to History Table

- We provide predicates to specify time-related condition as methods of HS_History type.
- When querying to history table,
  - You specify the name of history table ‘HS_TBL_<TableName>’ directly.
  - You specify the table obtained by invoking the method of HS_TYPE_<TableName> type.

History Types and Methods(1)

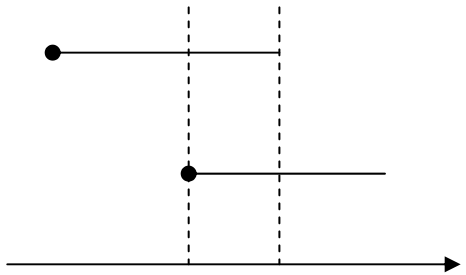
- Method provided in HS_History type:
  - Predicates:
    - HS_Overlaps: tests whether this period overlaps another period.
    - HS_Contains: tests whether this period contains another whole period.
    - HS_Meets: tests whether the end time of this period is equal to the begin time of another period or another TIMESTAMP value.
    - HS_Precedes: tests whether the whole of this period precedes the begin time of another period.
    - HS_Equals: tests whether this period is equal to another period.
  - Others:
    - HS_MonthInterval: returns the length of this period as year-month interval.
    - HS_DayInterval: returns the length of this period as day-time interval.
    - HS_Intersect: generates a new HS_History value with period which is overlap of this period and another period.

History Types and Methods(2)

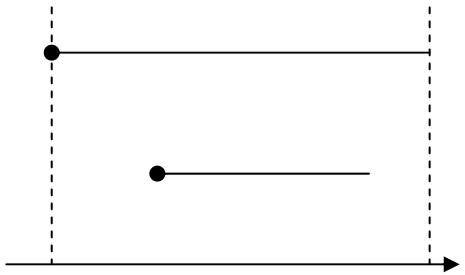
- Method provided in HS_TYPE_<TableName> type:
  - HS_HistoryBeginTime: sets the history begin time of the latest history row in the specified history row set.
  - HS_HistoryEndTime: sets the history end time of the latest history row in the specified history row set.
  - HS_Period: generates a restructured history table which stores change history of the specified values.

Time-Related Predicates

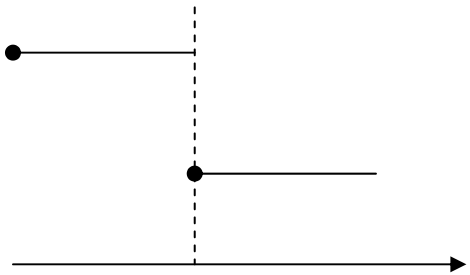
- The following figures express the meanings of the time-related predicates provided as methods of the HS_History type.



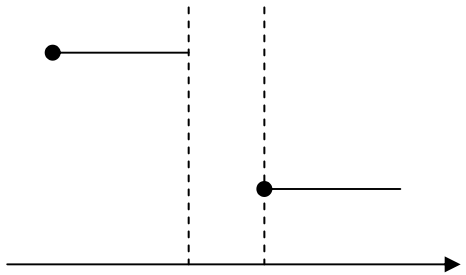
overlaps



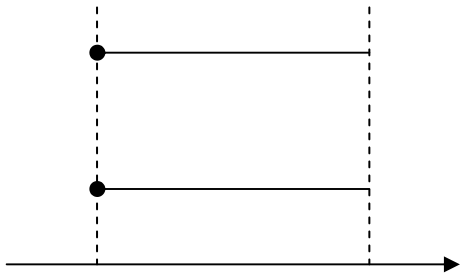
contains



meets



precedes



equals

Example: How to Store History Data

- There is a table called “emp”.
- History Table will be defined on top of this “emp” table.
- The next 11 slides will explain how History Table is managed.

There is a table called “emp” which is managing the employees of a university.

Table “emp”

EmpID	EmpName	Title	Salary	Dept
1	Tom	Assistant	4000	CS1

At 1996/04/03 09:32:17, the history table is created by calling HS_CreateHistory procedure.

Table “emp”

EmpID	EmpName	Title	Salary	Dept
1	Tom	Assistant	4000	CS1

```
CALL HS_CreateHistory(
  ‘emp’, ARRAY[‘Title’, ‘Salary’, ‘Dept’);
```

History table of “emp”

EmpID	EmpName	Title	Salary	Dept	HS_Hist	
					HS_HistoryBeginTime	HS_HistoryEndTime
1	Tom	Assistant	4000	CS1	1996-04-03 09:32:17	NULL

User invokes the HS_HistoryBeginTime method of HS_TYPE_emp type to update history begin time of the history rows to ‘1996-04-01 00:00:00’.

Table “emp”

EmpID	EmpName	Title	Salary	Dept
1	Tom	Assistant	4000	CS1

```
HS_TYPE_emp::HS_HistoryBeginTime(
   1,
   TIMESTAMP ‘1996-04-01 00:00:00’);
```

History table of “emp”

EmpID	EmpName	Title	Salary	Dept	HS_Hist	
					HS_HistoryBeginTime	HS_HistoryEndTime
1	Tom	Assistant	4000	CS1	1996-04-01 00:00:00	NULL

In subsequent slides, the process in which a user updates a value of history begin time after the CURRENT_TIMESTAMP value is stored in the value of history begin time is omitted for the simplification of explanation.

At 1997/04/01 00:00:00, Ken joined the faculty of this university.

Table “emp”

EmpID	EmpName	Title	Salary	Dept
1	Tom	Assistant	4000	CS1
2	Ken	Assistant Professor	7000	Med2

```
INSERT INTO emp(
      EmpID, EmpName, Title, Salary, Dept)
 VALUES(
      2, ‘Ken’, ‘Assistant Professor’, 7000, ‘Med2’);

HS_TYPE_emp::HS_HistoryBeginTime(
   2,
   TIMESTAMP ‘1997-04-01 00:00:00’);
```

History table of “emp”

EmpID	EmpName	Title	Salary	Dept	HS_Hist	
					HS_HistoryBeginTime	HS_HistoryEndTime
1	Tom	Assistant	4000	CS1	1996-04-01 00:00:00	NULL
2	Ken	Assistant Professor	7000	Med2	1997-04-01 00:00:00	NULL

At 1998/04/01 00:00:00, Ken was promoted from “assistant professor” to “professor”, and his salary was raised to 8,000.

Table “emp”

EmpID	EmpName	Title	Salary	Dept
1	Tom	Assistant	4000	CS1
2	Ken	Professor	8000	Med2

```
UPDATE emp
  SET Title = ‘Professor’, Salary = 8000
  WHERE EmpID = 2;

HS_TYPE_emp::HS_HistoryBeginTime(
    2,
    TIMESTAMP ‘1998-04-01 00:00:00’);
```

History table of “emp”

EmpID	EmpName	Title	Salary	Dept	HS_Hist	
					HS_HistoryBeginTime	HS_HistoryEndTime
1	Tom	Assistant	4000	CS1	1996-04-01 00:00:00	NULL
2	Ken	Assistant Professor	7000	Med2	1997-04-01 00:00:00	1998-04-01 00:00:00
2	Ken	Professor	8000	Med2	1998-04-01 00:00:00	NULL

At 1999/04/01 00:00:00, Tom's salary was raised to 5,000.

Table "emp"

EmpID	EmpName	Title	Salary	Dept
1	Tom	Assistant	5000	CS1
2	Ken	Professor	8000	Med2

```
UPDATE emp
  SET Salary = 5000
  WHERE EmpID = 1;

HS_TYPE_emp::HS_HistoryBeginTime(
    1,
    TIMESTAMP '1999-04-01 00:00:00');
```

History table of "emp"

EmpID	EmpName	Title	Salary	Dept	HS_Hist	
					HS_HistoryBeginTime	HS_HistoryEndTime
1	Tom	Assistant	4000	CS1	1996-04-01 00:00:00	1999-04-01 00:00:00
1	Tom	Assistant	5000	CS1	1999-04-01 00:00:00	NULL
2	Ken	Assistant Professor	7000	Med2	1997-04-01 00:00:00	1998-04-01 00:00:00
2	Ken	Professor	8000	Med2	1998-04-01 00:00:00	NULL

At 2000/04/01 00:00:00, Tom was promoted from “assistant” to “assistant professor”, and his salary was raised to 6,000.

Table “emp”

EmpID	EmpName	Title	Salary	Dept
1	Tom	Assistant Professor	6000	CS1
2	Ken	Professor	8000	Med2

```
UPDATE emp
  SET Title = ‘Assistant Professor’, Salary = 6000
  WHERE EmpID = 1;

HS_TYPE_emp::HS_HistoryBeginTime(
    1,
    TIMESTAMP ‘2000-04-01 00:00:00’);
```

History table of “emp”

EmpID	EmpName	Title	Salary	Dept	HS_Hist	
					HS_HistoryBeginTime	HS_HistoryEndTime
1	Tom	Assistant	4000	CS1	1996-04-01 00:00:00	1999-04-01 00:00:00
1	Tom	Assistant	5000	CS1	1999-04-01 00:00:00	2000-04-01 00:00:00
1	Tom	Assistant Professor	6000	CS1	2000-04-01 00:00:00	NULL
2	Ken	Assistant Professor	7000	Med2	1997-04-01 00:00:00	1998-04-01 00:00:00
2	Ken	Professor	8000	Med2	1998-04-01 00:00:00	NULL

At 2001/04/01 00:00:00, Ken was transferred to Med1 department.

Table “emp”

EmpID	EmpName	Title	Salary	Dept
1	Tom	Assistant Professor	6000	CS1
2	Ken	Professor	8000	Med1

```
UPDATE emp
  SET Dept = ‘Med1’
  WHERE EmpID = 2;

HS_TYPE_emp::HS_HistoryBeginTime(
    2,
    TIMESTAMP ‘2001-04-01 00:00:00’);
```

History table of “emp”

EmpID	EmpName	Title	Salary	Dept	HS_Hist	
					HS_HistoryBeginTime	HS_HistoryEndTime
1	Tom	Assistant	4000	CS1	1996-04-01 00:00:00	1999-04-01 00:00:00
1	Tom	Assistant	5000	CS1	1999-04-01 00:00:00	2000-04-01 00:00:00
1	Tom	Assistant Professor	6000	CS1	2000-04-01 00:00:00	NULL
2	Ken	Assistant Professor	7000	Med2	1997-04-01 00:00:00	1998-04-01 00:00:00
2	Ken	Professor	8000	Med2	1998-04-01 00:00:00	2001-04-01 00:00:00
2	Ken	Professor	8000	Med1	2001-04-01 00:00:00	NULL

At 2002/04/01 00:00:00, Ken was transferred to Med3 department.

Table “emp”

EmpID	EmpName	Title	Salary	Dept
1	Tom	Assistant Professor	6000	CS1
2	Ken	Professor	8000	Med3

```
UPDATE emp
  SET Dept = ‘Med3’
  WHERE EmpID = 2;

HS_TYPE_emp::HS_HistoryBeginTime(
    2,
    TIMESTAMP ‘2002-04-01 00:00:00’);
```

History table of “emp”

EmpID	EmpName	Title	Salary	Dept	HS_Hist	
					HS_HistoryBeginTime	HS_HistoryEndTime
1	Tom	Assistant	4000	CS1	1996-04-01 00:00:00	1999-04-01 00:00:00
1	Tom	Assistant	5000	CS1	1999-04-01 00:00:00	2000-04-01 00:00:00
1	Tom	Assistant Professor	6000	CS1	2000-04-01 00:00:00	NULL
2	Ken	Assistant Professor	7000	Med2	1997-04-01 00:00:00	1998-04-01 00:00:00
2	Ken	Professor	8000	Med2	1998-04-01 00:00:00	2001-04-01 00:00:00
2	Ken	Professor	8000	Med1	2001-04-01 00:00:00	2002-04-01 00:00:00
2	Ken	Professor	8000	Med3	2002-04-01 00:00:00	NULL

At 2003/04/01 00:00:00, Tom was transferred to CS2 department.

Table “emp”

EmpID	EmpName	Title	Salary	Dept
1	Tom	Assistant Professor	6000	CS2
2	Ken	Professor	8000	Med3

```
UPDATE emp
  SET Dept = ‘CS2’
  WHERE EmpID = 1;

HS_TYPE_emp::HS_HistoryBeginTime(
    1,
    TIMESTAMP ‘2003-04-01 00:00:00’);
```

History table of “emp”

EmpID	EmpName	Title	Salary	Dept	HS_Hist	
					HS_HistoryBeginTime	HS_HistoryEndTime
1	Tom	Assistant	4000	CS1	1996-04-01 00:00:00	1999-04-01 00:00:00
1	Tom	Assistant	5000	CS1	1999-04-01 00:00:00	2000-04-01 00:00:00
1	Tom	Assistant Professor	6000	CS1	2000-04-01 00:00:00	2003-04-01 00:00:00
1	Tom	Assistant Professor	6000	CS2	2003-04-01 00:00:00	NULL
2	Ken	Assistant Professor	7000	Med2	1997-04-01 00:00:00	1998-04-01 00:00:00
2	Ken	Professor	8000	Med2	1998-04-01 00:00:00	2001-04-01 00:00:00
2	Ken	Professor	8000	Med1	2001-04-01 00:00:00	2002-04-01 00:00:00
2	Ken	Professor	8000	Med3	2002-04-01 00:00:00	NULL

At 2004/04/01 00:00:00, Tom was transferred to another university.

Table “emp”

EmpID	EmpName	Title	Salary	Dept
2	Ken	Professor	8000	Med3

```
DELETE
  FROM emp
WHERE EmpID = 1;

HS_TYPE_emp::HS_HistoryEndTime(
   1,
   TIMESTAMP ‘2004-04-01 00:00:00’);
```

History table of “emp”

EmpID	EmpName	Title	Salary	Dept	HS_Hist	
					HS_HistoryBeginTime	HS_HistoryEndTime
1	Tom	Assistant	4000	CS1	1996-04-01 00:00:00	1999-04-01 00:00:00
1	Tom	Assistant	5000	CS1	1999-04-01 00:00:00	2000-04-01 00:00:00
1	Tom	Assistant Professor	6000	CS1	2000-04-01 00:00:00	2003-04-01 00:00:00
1	Tom	Assistant Professor	6000	CS2	2003-04-01 00:00:00	2004-04-01 00:00:00
2	Ken	Assistant Professor	7000	Med2	1997-04-01 00:00:00	1998-04-01 00:00:00
2	Ken	Professor	8000	Med2	1998-04-01 00:00:00	2001-04-01 00:00:00
2	Ken	Professor	8000	Med1	2001-04-01 00:00:00	2002-04-01 00:00:00
2	Ken	Professor	8000	Med3	2002-04-01 00:00:00	NULL

Examples of Queries to History Table of "emp"

- How much is the salary of each employee on July 1, 2001?
- Who was an assistant professor on October 1, 2001?
- Who was an assistant professor from May 1, 2000 to February 1, 2003?
- Who belonged to the Med2 Department continuously for over 2 years?
- Who has the longest period as an assistant professor?

Example of SQL Query(1)

- How much is the salary of each employee on July 1, 2001?

```
SELECT EmpName, Salary, HS_Hist
   FROM HS_TBL_emp
   WHERE HS_Hist.HS_Contains(
                TIMESTAMP'2001-07-01 00:00:00')
```

Answer:

EmpName	Salary	HS_Hist	
		HS_HistoryBeginTime	HS_HistoryEndTime
Tom	6000	2000-04-01 00:00:00	2003-04-01 00:00:00
Ken	8000	2001-04-01 00:00:00	2002-04-01 00:00:00

Example of SQL Query(2)

- Who was an assistant professor on October 1, 2001?

```
SELECT EmpName, Title, HS_Hist
    FROM HS_TBL_emp
    WHERE HS_Hist.HS_Contains(
                TIMESTAMP'2001-10-01 00:00:00')
            AND Title = 'Assistant Professor'
```

Answer:

EmpName	Title	HS_Hist	
		HS_HistoryBeginTime	HS_HistoryEndTime
Tom	Assistant Professor	2000-04-01 00:00:00	2003-04-01 00:00:00

Example of SQL Query(3)

- Who was an assistant professor from May 1, 2000 to February 1, 2003?

```
SELECT EmpName, Title, HS_Hist
    FROM HS_TBL_emp
    WHERE HS_Hist.HS_Overlaps(
                TIMESTAMP'2000-05-01 00:00:00',
                TIMESTAMP'2003-02-01 00:00:00')
            AND Title = 'Assistant Professor'
```

Answer:

EmpName	Title	HS_Hist	
		HS_HistoryBeginTime	HS_HistoryEndTime
Tom	Assistant Professor	2000-04-01 00:00:00	2003-04-01 00:00:00

Example of SQL Query(4)

- Who belonged to the Med2 Department continuously for over 2 years?

```
SELECT HT.EmpName, HT.Dept, HT.HS_Hist
   FROM TABLE(HS_TYPE_emp::HS_Period('Dept')) AS HT
   WHERE HT.Dept = 'Med2 '
      AND HT.HS_Hist.HS_MonthInterval() >= INTERVAL '2' YEAR
```

Answer:

EmpName	Dept	HS_Hist	
		HS_HistoryBeginTime	HS_HistoryEndTime
Ken	Med2	1997-04-01 00:00:00	2001-04-01 00:00:00

Example of SQL Query(5)

- Who has the longest period as an assistant professor?

```
SELECT HT1.EmpName, HT1.Title, HT1.HS_Hist
   FROM TABLE(HS_TYPE_emp::HS_Period('Title')) AS HT1
   WHERE HT1.Title = 'Assistant Professor '
      AND HT1.HS_Hist.HS_MonthInterval() = (
         SELECT MAX(HT2.HS_Hist.HS_MonthInterval())
            FROM TABLE(HS_TYPE_emp::HS_Period('Title')) AS HT2
            WHERE HT2.Title = 'Assistant Professor')
```

Answer:

EmpName	Title	HS_Hist	
		HS_HistoryBeginTime	HS_HistoryEndTime
Tom	Assistant Professor	2000-04-01 00:00:00	2004-04-01 00:00:00