

WG2 N1536
WG3: KOA-046

Temporal Features in SQL standard



Krishna Kulkarni,
IBM Corporation
krishnak@us.ibm.com
May 13, 2011

Agenda

- **Brief description of the SQL standard**
- **List of features in the latest version**
- **Application-time period tables**
- **System-versioned tables**
- **System-versioned application-time period tables**
- **Q & A**

Brief description of the SQL standard

- **ISO/IEC 9075, Database Language SQL is the dominant database language *de-jure* standard.**
- **First published in 1987. Revised versions published in 1989, 1992, 1999, 2003, and 2008.**
- **A new version of SQL expected in late 2011; likely to progress to FDIS stage after the current meetings.**
- **Multi-part standard – 9 Parts: 1, 2, 3, 4, 9, 10, 11, 13, and 14; Parts 3, 9, 10, and 13 are currently inactive.**
- **Large number of commercial implementations, multi-billion dollar industry.**
- **Large number of applications written using SQL.**

Brief description of the SQL standard (contd.)

- **Divided into 9 parts:**
 - Part 1 – Framework (SQL/Framework)
 - Part 2 – Foundation (SQL/Foundation)
 - Part 3 – Call-Level Interface (SQL/CLI)
 - Part 4 – Persistent Stored Modules (SQL/PSM)
 - Part 9 – Management of External Data (SQL/MED)
 - Part 10 – Object Language Bindings (SQL/OLB)
 - Part 11 – Information and Definition Schemas (SQL/Schemata)
 - Part 13 – SQL Routines and Types using the Java Programming Language (SQL/JRT)
 - Part 14 – XML-Related Specifications (SQL/XML)

Brief description of the SQL standard (contd.)

- **Part 2 – SQL/Foundation: Largest and the most important part SQL**
 - General-purpose programming constructs - Data types, expressions, predicates, etc.
 - Data Definition: CREATE/ALTER/DROP of tables, views, constraints, triggers, stored procedures, stored functions, etc.
 - Query constructs: SELECT, joins, etc.
 - Data Manipulation: INSERT, UPDATE, MERGE, DELETE, etc.
 - Access Control: GRANT, REVOKE etc.
 - Transaction Control: COMMIT, ROLLBACK, etc.
 - Connection Management: CONNECT, DISCONNECT, etc.
 - Session Management: SET SESSION statements.
 - Exception Handling: GET DIAGNOSTICS statement

Brief description of the SQL standard (contd.)

- **For conformance purpose, SQL is divided into a list of “Features”, grouped under two categories:**
 - Mandatory features
 - Optional features
- **To claim conformance, an implementation must conform to all mandatory features.**
- **An implementation may conform to any number of optional features.**
- **Both are listed in Annex F of each part of the SQL standard.**
- **SQL/Foundation:2008 specifies 164 mandatory features and 280 optional features.**

Agenda

- Brief description of the SQL standard
- List of features in the latest version
- Application-time period tables
- System-versioned tables
- System-versioned application-time period tables
- Q & A

List of features in the latest version

- **New version of SQL standard has completed the FCD ballot; expected to progress FDIS ballot in a couple of months.**
- **A total 34 new features have been added to SQL/Foundation– all in the optional category. This brings the list of optional features in SQL/Foundation to 314:**
 - F054 TIMESTAMP in DATE type precedence list
 - F202 TRUNCATE TABLE: identity column restart option
 - F314 MERGE Statement with DELETE branch
 - F383 Set column not null clause
 - F384 Drop identity property clause

List of features in the latest version (contd.)

- F385 Drop column generation expression clause
- F386 Set identity column generation clause
- F492 Optional table constraint enforcement
- F860 Dynamic <fetch row count> in <fetch first clause>
- F861 Top-level <result offset clause> in <query expression>
- F862 <result offset clause> in subqueries
- F863 Nested <result offset clause> in <query expression>
- F864 Top-level <result offset clause> in views
- F865 Dynamic <offset row count> in <result offset clause>
- F866 FETCH FIRST clause: PERCENT option
- F867 FETCH FIRST clause: WITH TIES option

List of features in the latest version (contd.)

- S401 Distinct types based on array types
- S402 Distinct types based on distinct types
- S403 ARRAY_MAX_CARDINALITY
- S404 TRIM_ARRAY
- T177 Sequence generator support: simple restart option
- T178 Identity columns: simple restart option
- **T180 System-versioned tables**
- **T181 Application-time period tables**
- T495 Combined data change and retrieval
- T521 Named arguments in CALL statement
- T522 Default values for IN parameters of SQL-invoked procedures

List of features in the latest version (contd.)

- T614 NTILE function
- T615 LEAD and LAG functions
- T616 NULL treatment option for LEAD and LAG functions
- T617 FIRST_VALUE and LAST_VALUE functions
- T618 NTH_VALUE function
- T619 Nested window functions
- T620 WINDOW clause: GROUPS option

Agenda

- Brief description of the SQL standard
- List of features in the latest version
- **Application-time period tables**
- System-versioned tables
- System-versioned application-time period tables
- Q & A

Feature T181 – Application-time period tables

- **Application-time period tables are tables that contain a PERIOD clause (newly-introduced) with an user-defined period name.**
- **Currently restricted to temporal periods only; may be relaxed in the future.**
- **Application-time period tables must contain two additional columns, one to store the start time of a period associated with the row and one to store the end time of the period.**
- **Values of both start and end columns are set by the users.**
- **Additional syntax is provided for users to specify primary key/unique constraints that ensure no two rows with the same key value have overlapping periods.**

Feature T181 – Application-time period tables (contd.)

- **Additional syntax is provided for users to specify referential constraints that ensure the period of every child row is completely contained in the period of exactly one parent row or in the combined period of two or more consecutive parent rows.**
- **Queries, inserts, updates and deletes on application-time period tables behave exactly like queries, inserts, updates and deletes on regular tables.**
- **Additional syntax is provided on UPDATE and DELETE statements for partial period updates and deletes.**

Creating an application-time period table

```
CREATE TABLE employees  
(emp_name VARCHAR(50) NOT NULL PRIMARY KEY,  
dept_id VARCHAR(10),  
start_date DATE NOT NULL,  
end_date DATE NOT NULL,  
PERIOD FOR emp_period (start_date, end_date),  
PRIMARY KEY (emp_name, emp_period WITHOUT OVERLAPS),  
FOREIGN KEY (dept_id, PERIOD emp_period) REFERENCES  
departments (dept_id, PERIOD dept_period);
```

Notes:

- PERIOD clause automatically enforces the constraint (*end_date* > *start_date*).
- The name of the period can be any user-defined name.
- The period is considered to start on the *start_date* value and end on the value just prior to *end_date* value. This corresponds to the (closed, open) model of periods.

Inserting rows into an application-time period table

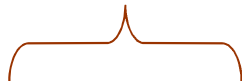
- **When a row is inserted into an application-time period table, user provides the start and end time of the period for each row.**
- **User-supplied time values can be either in the past, current, or in the future.**

Inserting rows into an application-time period table (contd.)

The following INSERT is executed in some transaction:

```
INSERT INTO employees (emp_name, dept_id, start_date, end_date)  
VALUES ('John', 'J13', DATE '1995-11-15', DATE '1996-11-15'),  
      ('Tracy', 'K25', DATE '1996-01-01', DATE '1997-11-15')
```

Business_time period
(closed, open)



emp_name	dept_id	start_date	end_date
John	J13	11/15/1995	11/15/1996
Tracy	K25	01/01/1996	11/15/1997

Business_time values are set by user.

Inserting rows into an application-time period table (contd.)

Given the following content

emp_name	dept_id	start_date	end_date
John	J13	11/15/1995	11/15/1996
Tracy	K25	01/01/1996	11/15/1997

The following INSERT will succeed

```
INSERT INTO employees (emp_name, dept_id, start_date, end_date)
VALUES ('John', 'J13', DATE '1996-11-15', DATE '1997-11-15'),
('John','J12', DATE '1997-11-15', DATE '1998-11-15')
```

While the following INSERT will not, because of the inclusion of *emp_period WITHOUT OVERLAPS* in the primary key definition:

```
INSERT INTO employees (emp_name, dept_id, start_date, end_date)
VALUES ('John', 'J13', DATE '1996-01-01', DATE '1996-12-31')
```

Updating rows in an application-time period table

- **All rows can be potentially updated.**
- **Users are allowed to update the start and end columns of the period associated with each row.**
- **When a row from an application-time period table is updated using the regular UPDATE statements, the regular semantics apply.**
- **Additional syntax is provided for UPDATE statements to specify the time period during which the update applies. Only those rows that lie within the specified period are impacted. May lead to row splits, i.e., update of a row may cause insertion of up to two rows to preserve the information for the periods that lie outside the specified period. Users are not allowed to update the start and end columns of the period associated with each row under this option.**

Updating rows in an application-time period table (contd.)

The following UPDATE is executed in some transaction:

```
UPDATE employees  
SET dept_id = 'J15'  
WHERE emp_name = 'John'
```

emp_name	dept_id	start_date	end_date
John	J15	11/15/1995	11/15/1996
Tracy	K25	01/01/1996	11/15/1997

No changes to the
Business_time values.

Updating rows in an application-time period table (contd.)

The following UPDATE is executed in some transaction:

```
UPDATE employees FOR PORTION OF emp_period FROM  
    DATE '1996-03-01' TO DATE '1996-07-01'  
SET dept_id = 'M12'  
WHERE emp_name = 'John'
```

emp_name	dept_id	start_date	end_date
John	J15	11/15/1995	03/01/1996
John	M12	03/01/1996	07/01/1996
John	J15	07/01/1996	11/15/1996
Tracy	K25	01/01/1996	11/15/1997

Automatic row splitting –
1 update and 2 inserts.

Deleting rows from an application-time period table

- **All rows can be potentially deleted.**
- **When a row from an application-time period table is deleted using the regular DELETE statements, the regular semantics apply.**
- **Additional syntax is provided for DELETE statements to specify the time period during which the delete applies. Only those rows that lie within the specified period are impacted. May lead to row splits, i.e., delete of a row may cause insertion of up to two rows to preserve the information for the periods that lie outside the specified period.**

Deleting rows from an application-time period table (contd.)

The following DELETE is executed in some transaction:

```
DELETE FROM employees FOR PORTION OF emp_period FROM  
DATE '1996-08-01' TO DATE '1996-09-01'  
WHERE emp_name = 'John'
```

emp_name	dept_id	start_date	end_date
John	J15	11/15/1995	03/01/1996
John	M12	03/01/1996	07/01/1996
John	J15	07/01/1996	08/01/1996
John	J15	09/01/1996	11/15/1996
Tracy	K25	01/01/1996	11/15/1997

Automatic row splitting –
1 delete and 2 inserts

Deleting rows from an application-time period table (contd.)

The following DELETE is executed in some transaction:

```
DELETE FROM employees  
WHERE EmpName = 'John'
```

emp_name	dept_id	start_date	end_date
Tracy	K25	01/01/1996	11/15/1997

All rows pertaining to John are deleted.

Querying an application-time period table

- **Existing syntax for querying regular tables is applicable to application-time period tables also.**

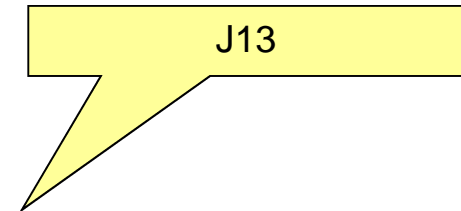
Querying an application-time period table (contd.)

employees

emp_name	dept_id	start_date	system_end
John	M24	1998-01-31	9999-12-31
John	J13	1995-11-15	1998-01-31
Tracy	K25	01/01/1996	2000-03-31

1. Which department was John in on Dec. 1, 1997?

```
SELECT dept_id  
FROM employees  
WHERE emp_name = 'John' AND start_date <= DATE '1997-12-01' AND  
end_date > DATE '1997-12-01';
```



Querying an application-time period table

employees

emp_name	dept_id	start_date	system_end
John	M24	1998-01-31	9999-12-31
John	J13	1995-11-15	1998-01-31
Tracy	K25	01/01/1996	2000-03-31

1. Which department is John in currently?

```
SELECT dept_id  
FROM employees
```

```
WHERE emp_name = 'John' AND start_date <= CURRENT_DATE AND end_date >  
CURRENT_DATE;
```



M24

Querying an application-time period table

employees

emp_name	dept_id	start_date	system_end
John	M24	1998-01-31	9999-12-31
John	J13	1995-11-15	1998-01-31
Tracy	K25	01/01/1996	2000-03-31

1. How many departments has John worked in since Jan. 1, 1996?

```
SELECT count(distinct dept_id)  
FROM employees WHERE emp_name = 'John' start_date <= CURRENT_DATE  
AND end_date > DATE '1996-01-01';
```

2

Benefits of application-time period tables

- **Most business data is time sensitive, i.e., need to track the time period during when a data item is deemed valid or effective from the business point of view.**
- **Database systems today offer no support for:**
 - Associating user-maintained time periods with rows.
 - Enforcing constraints such as “an employee can be in only one department in any given period”.
 - Updating/deleting a row for a part of its validity period.
- **Currently, applications take on the responsibility for managing such requirements. Major Issues:**
 - Complexity of code
 - Poor performance



Benefits of application-time period tables

- **Use of application-time period tables provides:**
 - Significant simplification of application code
 - Significant improvement in performance
 - Transparent to legacy applications

Agenda

- Brief description of the SQL standard
- List of features in the latest version
- Application-time period tables
- **System-versioned tables**
- System-versioned application-time period tables
- Q & A

Feature T180 - System-versioned tables

- **System-versioned tables are tables that contain a PERIOD clause with a pre-defined period name (SYSTEM_TIME) and specify WITH SYSTEM VERSIONING.**
- **System-versioned tables must contain two additional columns, one to store the start time of the SYSTEM_TIME period and one to store the end time of the SYSTEM_TIME period.**
- **Values of both start and end columns are set by the system. Users are not allowed to supply values for these columns.**
- **Unlike regular tables, system-versioned tables preserve the old versions of rows as the table is updated.**
- **Rows whose periods intersect the current time are called *current system rows*. All others are called *historical system rows*.**
- **Only current system rows can be updated or deleted.**
- **All constraints are enforced on current system rows only.**

Creating a system-versioned table

```
CREATE TABLE employees  
(emp_name VARCHAR(50) NOT NULL,  
dept_id VARCHAR(10),  
system_start TIMESTAMP(6) GENERATED ALWAYS AS ROW START,  
system_end TIMESTAMP(6) GENERATED ALWAYS AS ROW END,  
PERIOD FOR SYSTEM_TIME (system_start, system_end),  
PRIMARY KEY (emp_name),  
FOREIGN KEY (dept_id) REFERENCES departments (dept_id);  
) WITH SYSTEM VERSIONING;
```

Notes:

- PERIOD clause automatically enforces the constraint (*system_end* > *system_start*).
- The name of the period must be SYSTEM_TIME.
- The period is considered to start on the *system_start* value and end on the value just prior to *system_end* value. This corresponds to the (closed, open) model of periods.

Inserting rows into a system-versioned table

- **When a row is inserted into a system-versioned table, the SQL-implementation sets the start time to the transaction time and the end time to the largest timestamp value.**
- **All rows inserted in a transaction will get the same values for the start and end columns.**

Inserting rows into a system-versioned table (contd.)

The following **INSERT** is executed at timestamp 11/15/1995

```
INSERT INTO emp (emp_name, dept_id)
```

```
VALUES ('John', 'J13'), ('Tracy', 'K25')
```

employees

emp_name	dept_id	system_start	system_end
John	J13	1995-11-15	9999-12-31
Tracy	K25	1995-11-15	9999-12-31

System_start and system_end values are set by DBMS.

Notes:

- Only date components of *system_start* and *system_end* values are shown for the purpose of simplifying display.

Updating rows in a system-versioned table

- **When a row from a system-versioned table is updated, the SQL-implementation inserts the “old” version of the row into the table before updating the row.**
- **SQL-implementation sets the end time of the old row and the start time of the updated row to the transaction time.**
- **Users are not allowed to update the start and end columns.**

Updating rows in a system-versioned table (contd.)

The following **UPDATE** is executed at timestamp 1/31/1998 ...:

```
UPDATE emp
```

```
SET dept_id = 'M24'
```

```
WHERE emp_name = 'John'
```

employees

emp_name	dept_id	system_start	system_end
John	M24	1998-01-31	9999-12-31
John	J13	1995-11-15	1998-01-31
Tracy	K25	1995-11-15	9999-12-31

Deleting rows from a system-versioned table

- **When a row from a system-versioned table is deleted, the SQL-implementation does not actually delete the row; it simply sets its end time to the transaction time.**

Deleting rows from a system-versioned table (contd.)

The following **DELETE** is executed on 3/31/2000:

```
DELETE FROM emp
```

```
WHERE emp_name = 'Tracy'
```

employees

emp_name	dept_id	system_start	system_end
John	M24	1998-01-31	9999-12-31
John	J13	1995-11-15	1998-01-31
Tracy	K25	1995-11-15	2000-03-31

Querying system-versioned tables

- **Existing syntax for querying regular tables is applicable to system-versioned tables also.**
- **Additional syntax is provided for expressing queries involving system-versioned tables in a more succinct manner:**
 - FOR SYSTEM_TIME AS OF <datetime value expression>
 - FOR SYSTEM_TIME BETWEEN
 < datetime value expression 1> AND
 < datetime value expression 2>
 - FOR SYSTEM_TIME FROM
 < datetime value expression 1> TO
 < datetime value expression 2>

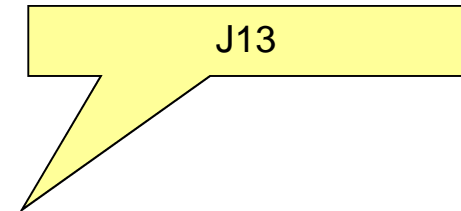
Querying system-versioned tables

employees

emp_name	dept_id	system_start	system_end
John	M24	1998-01-31	9999-12-31
John	J13	1995-11-15	1998-01-31
Tracy	K25	1995-11-15	2000-03-31

1. Which department was John in on Dec. 1, 1997?

```
SELECT Dept  
FROM employees FOR SYSTEM_TIME AS OF DATE '1997-12-01'  
WHERE emp_name = 'John'
```



Querying system-versioned tables

employees

emp_name	dept_id	system_start	system_end
John	M24	1998-01-31	9999-12-31
John	J13	1995-11-15	1998-01-31
Tracy	K25	1995-11-15	2000-03-31

1. Which department is John in currently?

```
SELECT Dept  
FROM employees  
WHERE emp_name = 'John'
```



M24

Note: If AS OF clause is not specified, only current system rows are returned, i.e., FOR SYSTEM_TIME AS OF CURRENT_TIMESTAMP is the default.

Querying system-versioned tables

employees

emp_name	dept_id	system_start	system_end
John	M24	1998-01-31	9999-12-31
John	J13	1995-11-15	1998-01-31
Tracy	K25	1995-11-15	2000-03-31

1. How many departments has John worked in since Jan. 1, 1996?

```
SELECT count(distinct dept_id)  
FROM employees FOR SYSTEM_TIME BETWEEN DATE '1996-01-01' AND  
CURRENT_DATE  
WHERE emp_name = 'John'
```

2

Benefits of system-versioned tables

- **Today's database systems focus mainly on managing current data; they provide almost no support for managing historical data.**
- **Some applications have an inherent need for preserving old data. Examples: job histories, salary histories, account histories, etc.**
- **Regulatory and compliance laws require keeping old data around for certain length of time.**
- **Currently, applications take on the responsibility for preserving old data. Major Issues:**
 - Complexity of code
 - Poor performance



Benefits of system-versioned tables (contd.)

- **Use of system-versioned tables provides:**
 - Significant simplification of application code
 - Significant improvement in performance
 - Transparent to legacy applications

Agenda

- **Brief description of the SQL standard**
- **List of features in the latest version**
- **Application-time period tables**
- **System-versioned tables**
- **System-versioned application-time period tables**
- **Q & A**

System-versioned application-time period tables

- **A table that is both an application-time period table and a system-versioned table.**
- **Such a table supports features of both application-time period tables and system-versioned tables.**

Creating a system-versioned application-time period table

```
CREATE TABLE employees
(emp_name VARCHAR(50) NOT NULL PRIMARY KEY,
dept_id VARCHAR(10),
start_date DATE NOT NULL,
end_date DATE NOT NULL,
system_start TIMESTAMP(6) GENERATED ALWAYS AS ROW START,
System_end TIMESTAMP(6) GENERATED ALWAYS AS ROW END,
PERIOD FOR emp_period (start_date, end_date),
PERIOD FOR SYSTEM_TIME (system_start, system_end),
PRIMARY KEY (emp_name, emp_period WITHOUT OVERLAPS),
FOREIGN KEY (dept_id, PERIOD emp_period) REFERENCES
    departments (dept_id, PERIOD dept_period)
) WITH SYSTEM VERSIONING;
```


Insert

On 11/01/1995, *employees* table was updated to show that John and Tracy will be joining the departments J13 & K25, respectively, starting from 11/15/1995.

```
INSERT INTO employees (emp_name, dept_id, start_date, end_date)  
VALUES ('John', 'J13', DATE '1995-11-15', DATE '9999-12-31'),  
      ('Tracy','K25', DATE '1995-11-15', DATE '9999-12-31')  
employees
```

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J13	11/15/1995	12/31/9999	11/01/1995	12/31/9999
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

system_start and system_end values are set by the system.

Note: DATE type is used in examples instead of TIMESTAMP type to simplify display.

Update

On 11/10/1995, it was discovered that John was assigned to the wrong department. It was changed to department J15 on that day.

```
UPDATE employees  
SET dept_id = 'J15'  
WHERE emp_name = 'John'
```

employees

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J15	11/15/1995	12/31/9999	11/10/1995	12/31/9999
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

Partial period update

On 12/15/1997, John is loaned to Dept M12 starting from 1/1/1998 to 7/1/1998.

*UPDATE employees FOR PORTION OF emp_period FROM
DATE '1998-01-01' TO DATE '1998-07-01'*

SET dept_id = 'M12' WHERE emp_name = 'John'

employees

Emp_name	dept_id	etart_date	end_date	system_start	system_end
John	J15	07/01/1998	12/31/9999	12/15/1997	12/31/9999
John	M12	01/01/1998	07/01/1998	12/15/1997	12/31/9999
John	J15	11/15/1995	01/01/1998	12/15/1997	12/31/9999
John	J15	11/15/1995	12/31/9999	11/10/1995	12/15/1997
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

Partial period delete

On 12/15/1998, John is approved for a leave of absence from 1/1/1999 to 1/1/2000.

*DELETE FROM employees FOR PORTION OF emp_period FROM
DATE '1999-01-01' TO DATE '2000-01-01'*

WHERE emp_name = 'John'

employees

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J15	01/01/2000	12/31/9999	12/15/1998	12/31/9999
John	J15	07/01/1998	01/01/1999	12/15/1998	12/31/9999
John	M12	01/01/1998	07/01/1998	12/15/1997	12/31/9999
John	J15	11/15/1995	01/01/1998	12/15/1997	12/31/9999
John	J15	07/01/1998	12/31/9999	12/15/1997	12/15/1998
John	J15	11/15/1995	12/31/9999	11/10/1995	12/15/1997
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

Delete

On 6/1/2000, John resigns from the company.

DELETE FROM employees

WHERE emp_name = 'John'

employees

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J15	01/01/2000	12/31/9999	12/15/1998	06/01/2000
John	J15	07/01/1998	01/01/1999	12/15/1998	06/01/2000
John	M12	01/01/1998	07/01/1998	12/15/1997	06/01/2000
John	J15	11/15/1995	01/01/1998	12/15/1997	06/01/2000
John	J15	07/01/1998	12/31/1999	12/15/1997	12/15/1998
John	J15	11/15/1995	12/31/9999	11/10/1995	12/15/1997
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

Agenda

- **Brief description of the SQL standard**
- **List of features in the latest version**
- **System-versioned tables**
- **Application-time period tables**
- **System-versioned application-time period tables**
- **Q & A**