SYBASE®

FAQ (Basic)

Sybase CEP Option R4

DOCUMENT ID: DC01023-01-0400-01
LAST REVISED: February 2010

Sybase, Inc., One Sybase Drive, Dublin, CA 94568

Contents

Preface

The *Frequently Asked Questions (Basics)* document contains answers to commonly asked questions about Sybase® CEP Engine, including Sybase CEP Server, the Continuous Computation Language (CCL), Sybase CEP Studio, and other related topics.

- For an overview of Sybase CEP Engine, please see the *Sybase CEP Technology Overview*.
- For Sybase CEP Server and Sybase CEP Studio installation and configuration instructions, please see the *Sybase CEP Installation Guide*.
- To get started with using Sybase CEP Engine, please see the *Sybase CEP Getting Started* tutorial.
- For more help with using Sybase CEP Studio interface, please see the *Sybase CEP Studio Guide*.
- For more advanced Sybase CEP Engine and CCL programming concepts, please see:
  - *Sybase CEP Frequently Asked Questions (Advanced)*.
  - *Sybase CEP Programmer's Guide*.
  - *Sybase CEP CCL Reference*.
  - *Sybase CEP Integration Guide*.

General Questions about Sybase CEP

Answers to general questions about the Sybase CEP Engine and its uses. Discusses Sybase CEP functionality, Continuous Computational Language, data streams, platform compatibility, and so on.

Questions

- What is Sybase CEP Engine?
- What kinds of problems does Sybase CEP Engine help to solve?
- What exactly is CCL?
- What is a data stream?
- How is Sybase CEP Engine different from Tibco, IBM MQ, or JMS-based publish-subscribe software?
- Is the Sybase CEP Engine a replacement for databases?
- Is the Sybase CEP Engine just an in-memory database?
- What if I still need to compare real-time data to historical data from my database, or to look up some reference information?
- How are CCL queries created and sent to Sybase CEP Engine?
- What platforms does Sybase CEP Engine run on? How is it implemented?
- How many messages per second can the Sybase CEP Engine handle?
- What is the processing latency of Sybase CEP Engine?
- What components does Sybase CEP Engine include?

What is Sybase CEP Engine?

Sybase CEP Engine is a software engine that can process and analyze tens and hundreds of thousands of messages per second. Sybase CEP Engine is not an end-user application, but rather an enabling technology and a platform to help build other applications. Sybase CEP Engine is easy to program, deploy, and scale. The key features of Sybase CEP Engine include:

- Sybase CEP Engine's ability to process data *continuously* as it arrives.
- Sybase CEP Engine's ability to process data before it is stored on disk, thus achieving very high throughput and a low latency of under one millisecond.
- The processing and correlation performed by the Sybase CEP Engine which is defined via a familiar-looking SQL-based Continuous Computation Language (CCL).

What kinds of problems does Sybase CEP Engine help to solve?

Problems ideal for Sybase CEP Engine generally have all or most of these properties:

- Rich data sources:
  - Multiple data streams (multiple data sources).
  - Heterogeneous data streams (multiple formats).
  - High incoming data rates.
- Complex processing requirements:
  - Processing requiring filtering, aggregation, transformation, multi-stream correlation, event pattern matching, and other complex processing.
- Need for immediate response:
  - Close to real-time response (latency requirements under one millisecond).

These problems are common in Finance (trading, risk management, fraud detection), Manufacturing (process control), Network Security (intrusion detection, event management), Supply Chain Management and Logistics (RFID applications), Power Grid Management, Sensor networks, Military, and other industries.

What exactly is CCL?

CCL is a language for continuously processing data arriving on multiple data streams. CCL uses familiar SQL constructs like **SELECT**, **FROM**, **WHERE**, **HAVING**, **GROUP BY**, **INSERT**, and others, to express queries over data streams. Unlike SQL "one time only" queries, CCL queries run continuously within the Sybase CEP Engine. Also, unlike a traditional database that runs SQL queries against mostly static data, Sybase CEP Engine runs continuously arriving data against pre-registered CCL queries.

What is a data stream?

A data stream is a way to *publish* and *subscribe* to data that needs to be continuously processed. Each stream row contains a number of columns, and each column has a name and type. For example:

- A stream called RFIDRaw, coming out of an RFID reader may have these columns:
  - ReaderID (**STRING**).

    - TagID (**STRING**).
- A stream called Trades, coming from a stock exchange may have columns:
    - Symbol (**STRING**).
    - Volume (**INTEGER**).
    - Price (**FLOAT**).

One way to think about data streams is to imagine them as database tables of infinite size. Queries referring to these "tables" typically operate not on the whole stream but on windows defined over portions of the stream. Some queries may write into the "table" and some may read from it. See *Stream Questions* on page 9 for more details about streams.

How is Sybase CEP Engine different from Tibco, IBM MQ, or JMS-based publish-subscribe software?

Sybase CEP Engine isn't just another "pub/sub" (publish and subscribe) product, like Tibco, or IBM MQ. Other products provide just a transport layer for data. They do not enable complex processing on the data they deliver. Sybase CEP Engine is first and foremost a rich processing platform, with an SQL-like processing and computation language. Sybase CEP Engine can also incorporate different types of message buses through the use of adapters.

Is the Sybase CEP Engine a replacement for databases?

No, it is not. Databases are great at storing and querying static data, and processing transactions reliably. However, databases are notoriously bad at processing continuously incoming large volumes of data with low latency. There are several reasons for this:

1. Databases must store all data on disk first before beginning to process it. This is slow.
2. Databases do not use pre-registered continuous queries. Database queries are "one time only" queries. If you want to ask a question ten times a second, you have to issue the query ten times a second. This model breaks down when there are many queries that need to be continuously evaluated.
3. Databases do not use incremental processing. Sybase CEP Engine is capable of evaluating queries in an incremental way as data arrives. Databases cannot do this.

Sybase CEP Engine complements traditional databases to help solve new classes of problems.

Is the Sybase CEP Engine just an in-memory database?

No. In-memory databases solve the problem of having to store data before processing it, but they do not address the need for continuous queries or incremental processing. Sybase CEP Engine has a highly optimized in-memory store to keep the state of the queries. CCL Query Module Persistence and Guaranteed Delivery features are used to ensure that no rows or state information are lost in the event of Sybase CEP Server failure by storing critical information about query module state on the disk drive. However, CCL queries also run continuously, addressing the need for ongoing incremental processing.

What if I still need to compare real-time data to historical data from my database, or to look up some reference information?

This is a very common problem that CCL was designed to solve. CCL makes it easy to write queries that continuously join data arriving on a data stream with data from a database. Sybase CEP Engine can cache whole database tables in memory, or it can bring in data required from a database subquery. You can also perform update and delete functions on an external database directly from CCL.

How are CCL queries created and sent to Sybase CEP Engine?

Sybase CEP provides an application called Sybase CEP Studio, which allows you to create CCL queries, organize them in hierarchical modules and projects, compile the projects, and deploy them to Sybase CEP Server. The use of Sybase CEP Studio is not required, however. Sybase CEP Engine can be used from the command line, using the c8_client utility. You can also embed Sybase CEP Engine in your product and provide your own GUI that generates CCL queries and sends them to Sybase CEP Engine via SOAP (Simple Object Access Protocol).

What platforms does Sybase CEP Engine run on? How is it implemented?

Sybase CEP Engine currently runs on Windows and on Linux and other UNIX-like operating systems, including Sun Solaris. Sybase CEP Engine is implemented as a highly optimized multithreaded C++ server, to provide maximum performance. Adapters for sending data to and receiving data from the Sybase CEP Engine can be implemented in C/C++, Java 1.5, .NET3, and other languages.

How many messages per second can the Sybase CEP Engine handle?

The throughput of the engine depends on the queries that it is processing. For simple queries, such as filtering, the performance of Sybase CEP Engine is measured in hundreds of thousands of messages per second. For more complex queries, including multiple joins over sliding windows, Sybase customers have timed the engine at tens of thousands of messages per second. But the exact numbers depend on the particular queries being run.

What is the processing latency of Sybase CEP Engine?

Sybase CEP Engine processing latency also depends on the query, but can be roughly estimated to be under one millisecond.

What components does Sybase CEP Engine include?

Sybase CEP Engine consists of the following components:

- Sybase CEP Server: The software engine that actually processes and correlates data streams at runtime.
- Sybase CEP Studio: The graphical environment for defining streams, adapters, CCL queries and CCL modules, as well as for managing CCL projects at runtime. The use of Sybase CEP Studio is optional for customers who use the c8_client command-line utility,

and for products that embed Sybase CEP Engine and can define and manage CCL modules using a SOAP (Simple Object Access Protocol) API.

- CCL Compiler: The compiler that translates and optimizes CCL queries for processing by Sybase CEP Server. The Compiler is invoked by Sybase CEP Studio or through an API.
- Input and Output Adapters: The components that translate different kinds of data to and from the format used by Sybase CEP Engine.
- Integration SDK: The set of tools for creating adapters (in C/C++, Java 1.5, .NET3, and so on) and for interfacing with the engine through SOAP (Simple Object Access Protocol) APIs.

Stream Questions

Answers to questions about streams, stream URIs, stream schemas, and related topics.

Questions

- What is a schema in Sybase CEP Engine?
- What is a stream URI?
- How does data get into Sybase CEP Engine?
- What kinds of data does Sybase CEP Engine produce?
- Can the Sybase CEP Engine process XML messages?

What is a schema in Sybase CEP Engine?

Just like an XML schema defines the format of an XML message, a schema in Sybase CEP Engine defines the format of a stream. A schema contains the names of the columns in the stream and their types. The schema itself is a simple XML file that can be created manually, or automatically by Sybase CEP Studio. Once the schema is defined, it is very easy to reuse it and associate new streams with the schema.

What is a stream URI?

Streams are a basic way to do pub/sub within Sybase CEP Engine environment. Each stream has a URI, of the form:

```
ccl://hostname:port/Stream/Workspace/Project/StreamName
```

This URI is all that's needed for someone to publish data to a stream and to subscribe to the stream to receive the published data.

The CCL: URI is a "logical" URI that identifies the stream regardless of the server in the cluster where the stream is currently hosted. The "physical" location of the stream is given by an http: URI, which can be viewed by right-clicking on the stream in Sybase CEP Studio and choosing Properties from the Sybase CEP Studio menu.

How does data get into Sybase CEP Engine?

Sybase CEP Engine has an extensive adapter architecture for converting many kinds of data to a normalized form. For example, an adapter may take data from a Tibco RV bus, or an RFID reader, or a syslog server, and convert it to CSF. Sybase CEP Engine builds some adapters, but also includes a very simple Adapter SDK that makes it easy to implement adapters in C/C++, Java 1.5, .NET3, and so on. CEP Engine can also access traditional historical data through the use of ODBC and JDBC adapters.

What kinds of data does Sybase CEP Engine produce?

The Sybase CEP Engine produces processed data, alerts, commands, and so on, in Common Stream Format (CSF). Just like there are input adapters for converting data into CSF, data produced in the Sybase CEP Engine is converted by output adapters before being sent to external dashboards in whatever format necessary. The converted data may then be stored in any database, or may be used by external applications for issuing commands or taking action (for example, to block an IP address in a firewall if an attack is discovered).

Can the Sybase CEP Engine process XML messages?

Yes, this is a very common use case. There are multiple ways to process XML messages in Sybase CEP Engine. Input and output adapters are available for reading and writing XML messages. Sybase CEP Engine includes an XML data type and a number of pre-defined functions specific to XML. These functions may be used to convert data between the XML data type and other CCL data types, extract values and elements from XML columns, produce rows based on aggregates of XML rows, create XML elements, append attributes or processing instructions to XML elements or append comments to XML trees, find XML values within XML elements, concatenate, insert, update or delete XML content, determine the values of XML expressions and apply XSL transformation to XML.

Basic CCL

Questions and answers to basic questions about CCL queries. Discusses what CCL queries loos like, filter queries, time-based windows, row-based windows, syntax rules, and so on.

Questions

- How is CCL different from SQL?
- What does the simplest CCL query look like?
- What does a simple filter query look like?
- What is a time-based window and how can it be used?
- What is a rows-based window?
- Can I use GROUP BY to process an incoming data stream, as if it were split into several streams, based on a specified attribute?
- How do I use the OUTPUT EVERY clause to produce output over regular time periods?
- How do I use the OUTPUT AFTER clause to delay output?
- Are there joins in CCL?
- What is a join between a stream and a window?
- What is a join between two windows?
- How do I do event pattern matching with CCL?
- How do I use data from traditional databases, like Oracle or MySQL, in CCL queries?
- How do I learn more about CCL and Sybase CEP Engine?

How is CCL different from SQL?

First and foremost, there is a difference in processing models: SQL queries are request-response queries and operate on tables. CCL queries are registered and continuously operate on data streams.

CCL also provides several extensions to SQL. These include an ability to abstract queries into hierarchies of parameterizable modules and projects, many kinds of windows that can be applied to data streams, **OUTPUT EVERY** / **OUTPUT AFTER** clauses, the ability to detect patterns of rows that occur or don't occur in one or more specified data streams, and several other extensions.

What does the simplest CCL query look like?

The following query takes all messages from InputStream and inserts them into OutputStream:

```
INSERT INTO OutputStream
SELECT *
FROM InputStream;
```

This is a continuous query. It continuously waits for data in InputStream. As soon as the data arrives, it is inserted into the OutputStream.

What does a simple filter query look like?

Here is an example of a simple filter:

```
INSERT INTO OutputStream
SELECT *
FROM InputStream
WHERE InputStream.Amount > 1000;
```

This query passes only those messages to OutputStream where the Amount column of InputStream has a value greater than 1000.

What is a time-based window and how can it be used?

Simple filter queries are stateless, in other words, they do not store any state. However, many queries need to store a certain state, such as a portion of the data stream or a value of an aggregator. These queries are defined in terms of the data window on a particular stream.

For example, suppose that a stock trading application needs to keep track of the highest price of a stock in the past 15 minutes. Here is a query to do that:

```
INSERT INTO MaxPrice
SELECT Trades.Symbol, MAX(Trades.Price)
FROM Trades KEEP 15 MINUTES
WHERE Trades.Symbol = 'IBM';
```

Note the `FROM Trades KEEP 15 MINUTES` part of the query. This defines a sliding 15 minute window over the Trades stream. This query produces output on every incoming message in Trades, as long as the value of the Symbol column is "IBM". The output does not include the current price of IBM, but the highest price of IBM within the 15 minute window.

Sybase CEP Engine keeps just enough data in memory to be able to produce the maximum value, and then discards old messages as they exit the window.

See also the discussion of rows-based windows, below, for more information about windows.

What is a rows-based window?

Suppose that you want to write a query similar to the one in the previous question, but instead of learning the highest price over the past 15 minutes, you want to know the price over the last fifteen trades. Then instead of using a time-based window, you would use a rows-based window:

```
INSERT INTO MaxPrice
SELECT Trades.Symbol, MAX(Trades.Price)
FROM Trades KEEP 15 ROWS
WHERE Trades.Symbol = 'IBM';
```

Rows-based windows are very versatile. For advanced features of rows-based windows, see the Sybase CEP *Frequently Asked Questions (Advanced)* document or the Sybase CEP *Programmer's Guide*.

Can I use GROUP BY to process an incoming data stream, as if it were split into several streams, based on a specified attribute?

Yes, this is a very powerful feature of CCL. For example, previous questions keep the highest price of IBM over some time period. But what if we want to keep the price of all stocks in the stream over the past 15 minutes? This is where **GROUP BY** may be used. Here is an example:

```
INSERT INTO MaxPrice
SELECT Trades.Symbol, MAX(Trades.Price)
FROM Trades KEEP 15 MINUTES
GROUP BY Trades.Symbol;
```

The **GROUP BY** clause here tells CCL that this query should keep separate states for all trades with different symbols. Logically, the stream is split into a set of streams according to the symbol. The resulting streams are merged back after processing, so that the output stream

is again one stream that lists the maximum value of each stock over the last 15 minutes. There is one output value in MaxPrice for every input value in Trades. **GROUP BY** is quite powerful and can be used to split streams by IP address, Employee ID, Event ID, RFID Tag ID, Reader ID, or any other column.

How do I use the OUTPUT EVERY clause to produce output over regular time periods?

In general, CCL queries produce results whenever there are new messages in the input stream (see Sybase CEP *Frequently Asked Questions (Advanced)* for more details). However, sometimes it is a good idea to stabilize the stream: that is to produce data at predetermined time intervals, for example, every minute. **OUTPUT EVERY** is designed for this. Look at the following query:

```
INSERT INTO MaxPrice
SELECT Trades.Symbol, Trades.Price,
MAX(Trades.Price)
FROM Trades KEEP 15 MINUTES
GROUP BY Trades.Symbol
OUTPUT EVERY 1 MINUTE;
```

This query will produce output once a minute, for those stocks where there was activity during that minute. To see outputs of the last value for stocks that had no activity, use the **OUTPUT ALL EVERY** clause.

How do I use the OUTPUT AFTER clause to delay output?

Sometimes it is necessary to delay all output from a query by some time period. For example, this is useful when setting up a timer based on each message. **OUTPUT AFTER** is used in this case. Here is an example:

```
INSERT INTO MaxPrice
SELECT Trades.Symbol, Trades.Price, MAX(Trades.Price)
FROM Trades
KEEP 5 MINUTES
GROUP BY Trades.Symbol
OUTPUT AFTER 15 MINUTES;
```

OUTPUT AFTER delays the entire stream, or shifts the time series by a specified time period.

Are there joins in CCL?

There are several kinds of joins in CCL. There is a join between a stream and one or more windows, and also a join between multiple windows. CCL also supports both inner and outer joins. Finally, there are joins between a stream and data coming from a traditional relational database.

What is a join between a stream and a window?

Suppose that every time an event happens in stream S1, you want to know whether an event also happened in stream S2 within the past 30 minutes, in all cases where the Id column in S1 and the Id column in S2 have the same value. This is the simplest illustration of a join between a stream and a window. It looks like this:

```
INSERT INTO Result
SELECT *
FROM
S1,
S2 KEEP 30 MINUTES
WHERE
S1.Id = S2.Id;
```

Just as with a database join, this join performs a cartesian product between S1 and S2, subject to the join condition `S1.id=S2.id` . Unlike a database join, however, this join is continuous, in other words, it continuously publishes new rows into the Result stream as rows arrive in S1. The join remembers 30 minutes worth of rows in S2, so that it can compare incoming S1 values against stored S2 values.

In the case of stream-window joins, only rows arriving in the stream, in this case S1, trigger new output rows. Rows arriving in S2 do not result in new output rows. See the Sybase CEP *Frequently Asked Questions (Advanced)* document for more information on this subject.

What is a join between two windows?

Suppose that you have streams S1 and S2 (just like in the previous example) but whenever an event occurs in either stream, you want to know whether a corresponding event occurred in the other stream. In this case, you can join two windows:

```
INSERT INTO Result
SELECT *
```

```
FROM
S1 KEEP 30 MINUTES,
S2 KEEP 30 MINUTES
WHERE
S1.id = S2.id;
```

This is the simplest case of a join between two windows. The windows may be of different kinds, durations, and so on. The **WHERE** condition used to filter query results may be arbitrarily complex.

How do I do event pattern matching with CCL?

CCL comes with a powerful event pattern matching syntax, introduced with the keyword **MATCHING**. For example, suppose that you have streams S1 and S2, and you want to be notified if two related events occur in S1 within a ten-second interval, but no event occurs in S2 during this time. Here is a query that watches for this pattern. In this example, the relationship between S1 and S2 is established based on column Id:

```
INSERT INTO Alerts
SELECT *
FROM
S1 AS a,
S1 AS b,
S2 AS c
MATCHING [10 SECONDS: a, b, !c]
ON a.Id = b.Id = c.id;
```

Note several aspects of this query:

- The syntax of the query is somewhat similar to a join, but the windows are defined implicitly by the **MATCHING** clause (which is set to ten (10) seconds, in this case).
- The "sequence" operator, designated by a comma (,) is used to denote one event happening after another in the stated order.
- The "not" operator, designated by an exclamation point (!) is used to specify a non-event, in other words, an event that does *not* occur within the specified time period.
- The **ON** clause is used to provide a condition that relates the events.

Just as with a join, a **WHERE** clause may also be used to provide extra conditions.

How do I use data from traditional databases, like Oracle or MySQL, in CCL queries?

CCL lets you embed SQL statements into CCL queries. SQL queries are executed on demand against the target database, and the results are cached in memory, if necessary. Here is a sample query to illustrate a database subquery:

```
INSERT INTO OutHistoricalComparison
SELECT InTrades.Symbol, InTrades.Price, DbResult.ClosingPrice
FROM
  InTrades,
  (DATABASE 'OracleDb'
   SCHEMA '../schemas/closing-price.ccs'
   [[SELECT closing_price
     FROM Price_history ph
     WHERE ph.Symbol = ?InTrades.Symbol
     AND ph.closing_date = current_date-1]]
  ) as DbResult
;
```

This query is a join (or a correlated subquery) that joins the InTrades stream with an Oracle table called Price_history. Note the following characteristics of this query:

- The Oracle SQL query is enclosed in double brackets ([[...]]) to clearly separate it from the CCL query. This syntax is required.
- The SQL query refers to data from the stream column InTrades.Symbol to perform the join (note the question mark to indicate data from the Sybase CEP data stream).
- The database connection `'OracleDb'` must first be defined in a configuration file called `c8-services.xml` before being used in the database subquery.
- Data from the database will be cached, according to a cache policy specified in the same file.

How do I learn more about CCL and Sybase CEP Engine?

For more information, please see the following documents:

- *Sybase CEP Getting Started* tutorial.
- *Sybase CEP Frequently Asked Questions (Advanced).*
- *Sybase CEP Studio Guide.*
- *Sybase CEP Continuous Computation Language Programmer's Guide.*

How do I learn more about CCL and Sybase CEP Engine?

- *Sybase CEP CCL Reference.*
- *Sybase CEP Integration Guide.*
- *Sybase CEP Examples Guide.*