



Dealing with SQL Width in the Progress OpenEdge RDBMS:

# **AUTONOMOUS SCHEMA UPDATE (ASU)**

As a continuation of our efforts to deliver extensive SQL enhancements to Progress® OpenEdge® 11, customers will now find two new features to help better manage issues that may arise due to SQL width. The SQL width parameters cause truncation in data that impacts the accuracy and ability to run complete query results sets. To assist database administrators in addressing the issue of SQL width, we are presenting, as a series two complementary technical whitepapers each detailing a specific solution: [Dealing with SQL Width in the Progress OpenEdge RDBMS: Authorized Data Truncation \(ADT\)](#), and [Dealing with SQL Width in the Progress OpenEdge RDBMS: Autonomous Schema Update \(ASU\)](#).

## Overview

[Progress OpenEdge 11.6](#) contains a new SQL feature called Autonomous Schema Update (ASU). This feature helps overcome the SQL width problem that is encountered when database column values are larger than the column's size as defined in SQL. It solves this problem by automatically updating the SQL column width in schema wherever there is a data truncation event for the column. Autonomous Schema Update is based on truncation events generated by Authorized Data Truncation (ADT). (For more information on Authorized Data Truncation, please refer to this recently updated [whitepaper](#).)

ASU can prevent queries from failing when large column values are read. As ASU is dependent on truncated events generated by ADT, switching on ASU turns on ADT internally. By default, if the width of the column data being operated on in a query exceeds the defined width of the column, SQL returns an error, and as a result, the query fails. If ASU is enabled, SQL will instead truncate the large value down to the defined size and updates the offending column width to the highest in the result set. Due to column width update, subsequent execution of the query returns complete data instead of truncated data. Thus, ASU will permanently update the schema automatically without any user intervention and at the appropriate time.

# Description

Though SQL throws an error while operating on large data, it is now possible to authorize SQL to update schema while operating on large data using ASU. Autonomous Schema Update supports only varchar column type. This includes support for varchar arrays as well (both fixed and var arrays). There is a new startup parameter for database server with which we can enable or disable ASU. The new startup parameter is “-SQLWidthUpdate.”

## Server Startup Parameter

The following syntax is used to authorize SQL to update schema of large data columns during server startup:

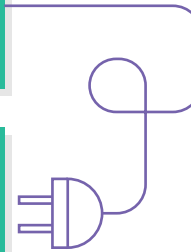
```
proserve -db <dbname> -S <port-number> -SQLWidthUpdate <on/off>
```

ASU parameter can only be provided during server startup. Unlike ADT, ASU parameter can't be provided in connection URL.

1. The “-SQLWidthUpdate” value is remembered for the lifetime of the server for all connections.
2. SQL will truncate data exceeding the column size if “-SQLWidthUpdate” is set to “on.” This truncation is only for the first execution of the query. SQL updates the width of large data columns to the highest value in the result set at this time. Subsequent execution of the query will provide complete data as schema of large columns is updated.
3. SQL will return an error in case of data exceeding the column size if “-SQLWidthUpdate” is set to “off.”
4. SQL returns an error in case of data exceeding the column size if the value of “-SQLWidthUpdate” is not set.
5. The value of “-SQLWidthUpdate” is logged in the database lg file (irrespective of whether this optional parameter is specified or not).

## What happens when ASU is set?

1. SQL turns on ADT internally. To be more precise, ASU sets ADT to “all.”
2. SQL returns truncated results for the first execution of the query. At the same time, SQL updates schema of large data columns in the query to the highest length in the result set.
3. Subsequent execution of the query results in complete data as schema of the large data columns is updated.



## Example

For example, let's assume that the following table is created in the database:

```
Create table pub.myTab (name varchar (5));
```

The column name has an entry “Progress Software” and “TomCruise” which exceeds the size of the column i.e., five.

Output of the query “select \* from pub.myTab” when ASU is on:

Progr

TomCr



At this point of time, SQL updates the width of “name” column to the highest width in the result set. In the current result set, highest width is for column value “Progress Software.” Length of this column value is 17. Hence, ASU updates the width of the “name” column to 17.

When the same query is executed once again, the output will be (provided that there are no INSERTs of data of length higher than 17):

Progress Software  
TomCruise

The following steps summarize what happens when ASU is set with SQL queries:

```
--Check column widths
SELECT COL, WIDTH FROM SYSPROGRESS.SYSCOLUMNS WHERE TBL='myTab'
ORDER BY 1,2;
```

COL	WIDTH
NAME	5

```
--First execution of the query
SELECT * FROM PUB.myTab;
```

NAME
Progr
TomCr

```
--Check column widths once again to see updated width
SELECT COL, WIDTH FROM SYSPROGRESS.SYSCOLUMNS WHERE TBL='myTab'
ORDER BY 1,2;
```

COL	WIDTH
NAME	17

```
--Subsequent executions of the query results in complete data as
ASU updated the schema
SELECT * FROM PUB.myTab;
```

NAME
Progress Software
TomCruise

## ASU and ADT Options

When ASU is turned on ADT is also enabled. The default for ADT is set to “All.” ADT has four options: All, Output, Off and Blank. ADT can be set at the server or connection level.

As ASU works on truncation events generated by ADT, it is very important to understand the various possible combinations of ASU and ADT. Moreover, as ADT option can also be specified at connection level, the combinations can sometimes become complex to setup and understand. The following table illustrates all the possible combinations of ASU and ADT options at all the possible levels. This table can be handy to clearly understand and set required options at required level.

(Please refer to the whitepaper on ADT for additional details on these options.)

USE CASE	ASU value (-SQLWidthUpdate)	ADT value - server level (-SQLTruncateTooLarge)	ADT value - connection level (truncateTooLarge)	OUTPUT*	SCHEMA UPDATE?
1	ON	ALL	ALL	Truncated data for the first run Complete data from next run	●
2	ON	ALL	OUTPUT		●
3	ON	ALL	NOT SPECIFIED		●
4	ON	OFF	ALL		●
5	ON	OFF	OUTPUT		●
6	ON	OFF	NOT SPECIFIED		●
7	ON	OUTPUT	ALL		●
8	ON	OUTPUT	OUTPUT		●
9	ON	OUTPUT	NOT SPECIFIED		●
10	ON	NOT SPECIFIED	ALL		●
11	ON	NOT SPECIFIED	OUTPUT		●
12	ON	NOT SPECIFIED	NOT SPECIFIED		●
13	ON	ALL	OFF	Value exceeding its max length or precision error	NO
14	ON	OFF	OFF		NO
15	ON	OUTPUT	OFF		NO
16	ON	NOT SPECIFIED	OFF		NO
17	OFF	ALL	OFF		NO
18	OFF	OUTPUT	OFF		NO
19	OFF	OFF	OFF		NO
20	OFF	OFF	NOT SPECIFIED		NO
21	OFF	NOT SPECIFIED	OFF		NO
22	OFF	NOT SPECIFIED	NOT SPECIFIED		NO
23	OFF	ALL	ALL	Truncated data for all the runs	NO
24	OFF	ALL	OUTPUT		NO
25	OFF	ALL	NOT SPECIFIED		NO
26	OFF	OFF	ALL		NO
27	OFF	OFF	OUTPUT		NO
28	OFF	OUTPUT	ALL		NO
29	OFF	OUTPUT	OUTPUT		NO
30	OFF	OUTPUT	NOT SPECIFIED		NO
31	OFF	NOT SPECIFIED	ALL		NO
32	OFF	NOT SPECIFIED	OUTPUT		NO

Table 1 – Various possible ASU and ADT combinations

\* - Output is based on assumption that there are no INSERT statements between first query execution and the second query execution.

As we see in the table, ASU and ADT options can be set at different levels (server and connection levels). While ASU can be set only at server level, ADT can be set at both server and connection levels. In the case where ADT is set at both server and connection levels, option set at connection level always takes precedence. For example, let's consider a case where ADT is set to "all" at server level and "off" at connection level. In this case, ADT is set to "off" which is set at connection level.

Let us understand the first entry of the table for instance. ASU value and ADT values at both server and connection level is set to "all" in this case. As ASU is on, first execution of the SQL query containing large column values return truncated results. At the same time, schema of the table is updated to the highest length in the result set. This makes subsequent execution of the query to generate complete data instead of error or truncated data.

Similarly, let's consider the fourth case. In this case, ASU is set to "all," ADT at server level is set to "off" and ADT at connection level is set to "all." SQL in this case also generates truncated results for the first execution of the query and then updates the schema. This makes the subsequent execution of the query to generate complete data in the result instead of error or truncated data. This is due to the fact that ASU internally switches ADT to "all" at server level. This happens even when the user explicitly makes ADT value as "off" at server level.

Another important case is where ASU is set to "all" but ADT is set to "off" at connection level. This means that the connection is explicitly mentioning that it cannot have truncated results. As specified in previous sections, ASU works on truncation events generated by ADT. Setting ADT to "off" stops SQL generating truncation events. Hence, in this case, ASU cannot make any schema update. Output of the query in this case always shows exceeding max width or precision error.

In the cases where ASU is set to "off," the query always generates a truncated result when ADT is "all/output." Schema update cannot happen in these cases as ASU is set to "off."

# Summary

Authorized Data Truncation provides the ability to return truncated data from a SQL query. Autonomous Schema Update provides the ability to have the database update the SQL schema width behind the scenes, so that the SQL query that is run against the same data set provides a complete result set that is not truncated. This facilitates better long-term health of the database and ensures more accurate query results. Although periodic scanning and maintenance are still required, the combination of these features eliminates the daily burden from the DBA to actively monitor and maintain SQL field lengths.

For more OpenEdge technical tips and tricks, visit [Progress Community](#).




## About Progress

Progress (NASDAQ: PRGS) is a global leader in application development, empowering the digital transformation organizations need to create and sustain engaging user experiences in today's evolving marketplace. With offerings spanning web, mobile and data for on-premise and cloud environments, Progress powers startups and industry titans worldwide, promoting success one customer at a time. Learn about Progress at [www.progress.com](http://www.progress.com) or 1-781-280-4000..

## Worldwide Headquarters

Progress, 14 Oak Park, Bedford, MA 01730 USA Tel: +1 781 280-4000 Fax: +1 781 280-4095

On the Web at: [www.progress.com](http://www.progress.com)

Find us on  [facebook.com/progresssw](https://facebook.com/progresssw)  [twitter.com/progresssw](https://twitter.com/progresssw)  [youtube.com/progresssw](https://youtube.com/progresssw)

For regional international office locations and contact information, please go to [www.progress.com/worldwide](http://www.progress.com/worldwide)

Progress and Progress OpenEdge are trademarks or registered trademarks of Progress Software Corporation and/or one of its subsidiaries or affiliates in the U.S. and/or other countries. Any other trademarks contained herein are the property of their respective owners.

© 2016 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.  
Rev 16/12 | 161118-0109

