MULTI-TENANCY INTRODUCTION AND OVERVIEW

John Sadd
Fellow and OpenEdge Evangelist
Document Version 1.0
November 2011

DISCLAIMER

This document accompanies the first in a series of short videos to introduce you to the new support in OpenEdge 11 for multi-tenant databases. By the end of this paper, I will have shown you how to create a new database for OpenEdge 11 and enable it for multi-tenancy, and give you a look at the new **Database Administration Console** tool, which supports your work with multi-tenant databases.

But before that, I'll spend a few minutes explaining what multi-tenancy is all about. There's a fair amount of terminology that I'll introduce along the way, but in this first paper I just try to explain: *What are tenants, anyway?*

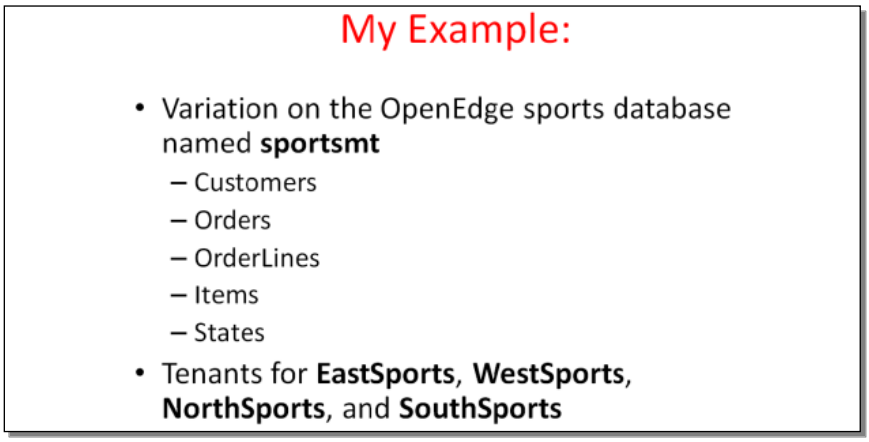
Tenants are named groups of people, users of one or more applications, who are related in some organizational way. The idea of tenants becomes significant only when there are two or more such groups that can be distinguished in some way. Each group shares data specific to its organization, and it's important to keep that data distinct from the data used by other tenants. But all tenants run the same application or applications. An example could be the different automotive brands used in the Auto Edge sample application:

Car dealers for these different brands and their customers might all need the same kind of data, and the same application support for the sales effort, but of course each group of dealers for a given brand needs to keep their data separate from other groups. If you are an application vendor supporting multiple customers like these car brands by selling software as a service, you could make each car brand a tenant in a single database.

Or the manufacturers and distributors of various kinds of beer might run the same application, but they would need to keep their data securely separated from their competitors:

Again, as a SaaS provider, you could make each customer a tenant in a single multi-tenant database.

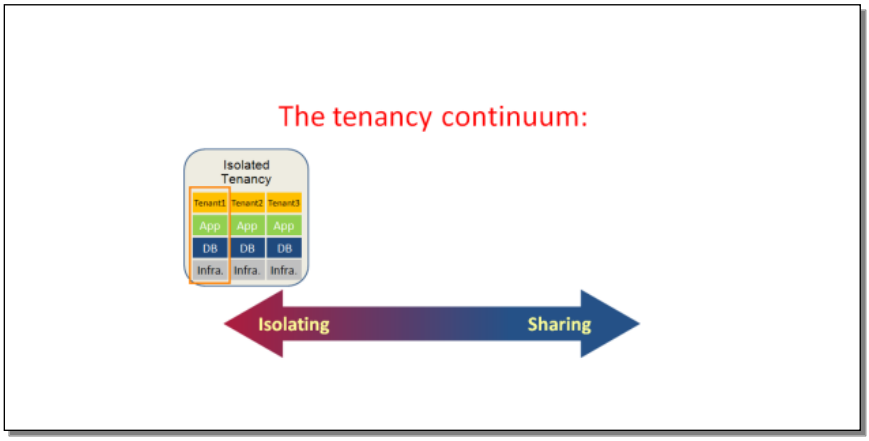
In this series of introductory videos and papers I use a somewhat simplified version of the familiar sports database, which I call **sportsmt**:



The back story for this example is that my company is a SaaS provider. We sell software as a service. We have an application we've sold to multiple organizations, but we support them by hosting their database and the rest of the hardware and software infrastructure at our own facility. We've been using four separate instances of the database and the application to support four different groups of customers, called East, West, North, and SouthSports. Now the goal is to combine these

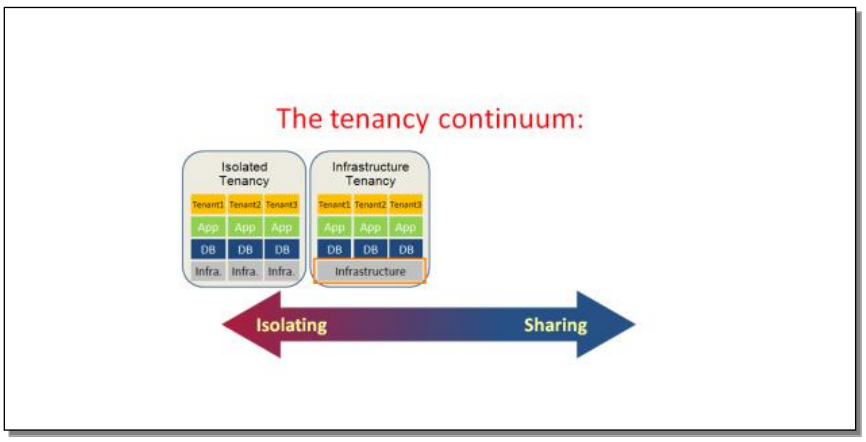
databases and make each of these customer groups into a tenant in that one new database.

So why would I want to do this? We can examine the different approaches to supporting multiple customers with distinct needs along what we call the tenancy continuum. At one extreme is a configuration we call **isolated tenancy**:

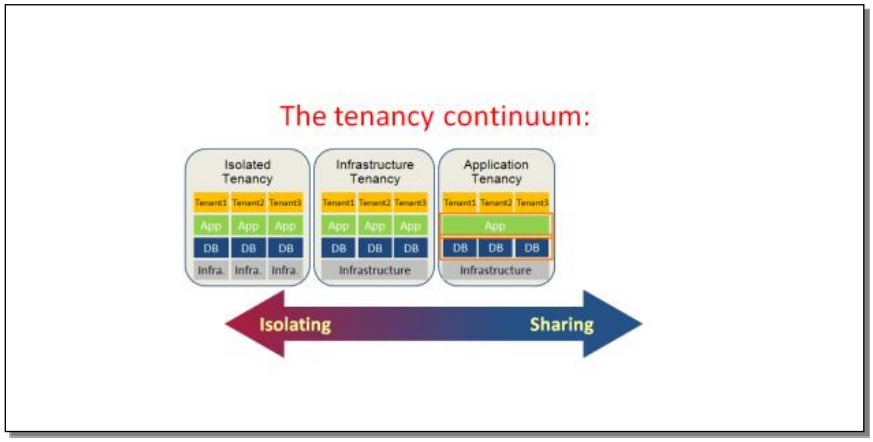


In this setup every tenant runs its own copy of the application. There may be customizations for each different customer group that are coded into the different code bases for the application. Each one has its own database as well. The schema definitions for the different customers might be the same, or there might be slight differences because of customizations, but the databases are distinct to keep the data separate. And I presume that underneath, they might even be running different versions of the OpenEdge product, or on different hardware and operating system versions. So everything is distinct, isolated, for each organization.

Infrastructure tenancy takes one step towards unifying everything:

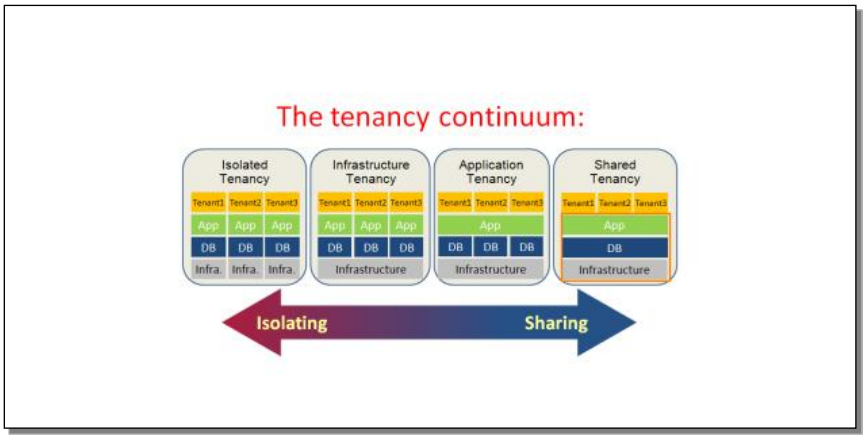


Here we just presume that each database and application code base is running on the same underlying hardware and software platform, including the same version of OpenEdge. As we continue along the line, the next step is **application tenancy**:



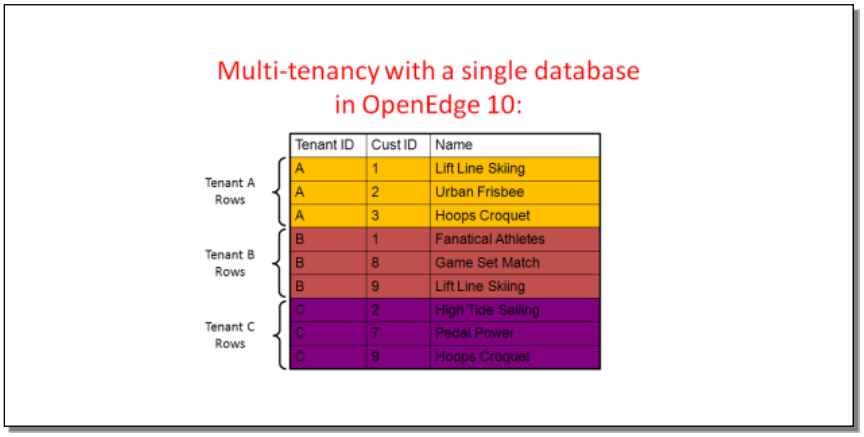
In this case we've integrated any customizations into a single version of the application code base, so everyone is running the same application, but each organization still has its own database. So far these are all configuration choices that you have control over. You can run everyone on the same hardware and software if you want. You can also code customizations into a single application code base if you want.

But there can be advantages to combining the data into a single instance of the database, which can greatly reduce your maintenance effort, the amount of hardware you need, and in other ways. We call this **shared tenancy**:



On the one hand, isolating everything makes it easier to customize the application for different customers, and set up security for them individually with no danger of anyone accidentally accessing the wrong data. You can adjust the level of performance to satisfy different SLAs -- service-level agreements -- and this kind of configuration can be a natural outgrowth of a situation where you start out just supporting one customer at a time. On the other hand, greater sharing gives you economy of scale, fewer hardware and software resources, and perhaps fewer personnel and lower management costs, to support a growing set of customers. To the extent that you have a large number of customers who need more or less the same application and service support, it's easier to support and manage them together.

But the shared tenancy scenario is hard to pull off in OpenEdge 10. In order to combine data for different customers, what we'll now call different **tenants**, in a single database, you've got to have a column in each affected table that identifies the tenant, either an explicit **tenant id**, or some other column that identifies the tenant, such as a customer number or name, or a division name within a company, like this:



Then you have to make absolutely sure that everywhere in your application where a user retrieves or updates data, that data is filtered by the tenant id. This is obviously a cumbersome approach, involving lots of application code changes, and it's just inherently unreliable. You're almost certain to miss something along the way.

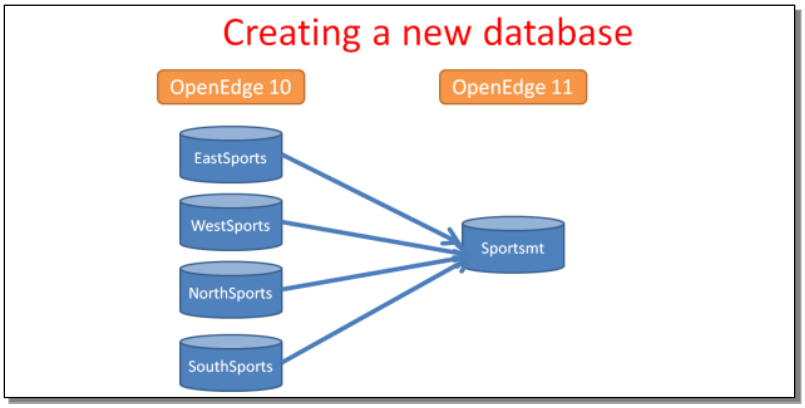
What you want is to avoid this reliance on an extraneous tenant identifier, so you don't have to make schema changes to your existing database, and you don't need to try to locate every place in the application where someone accesses data that is different for each tenant:

You just want to say, effectively, **`FOR EACH Customer`**, and always get just your customers, whichever organization you belong to.

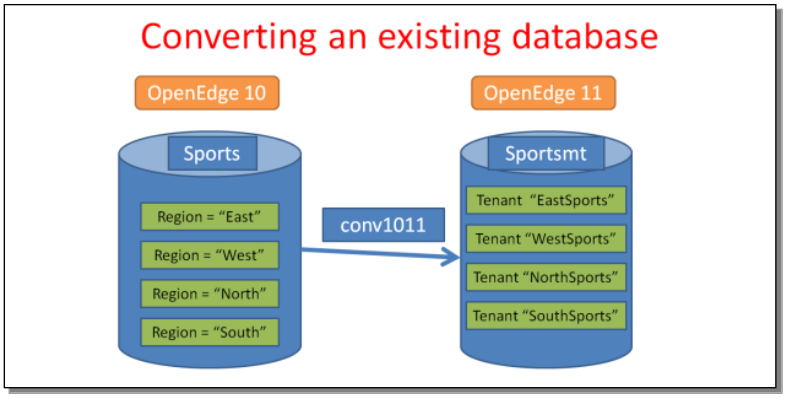
This is exactly what we've done in OpenEdge 11. We've built multi-tenancy right into the database, so you don't have to worry about it. Data for each tenant is physically partitioned separately, so there's no danger of anyone retrieving the wrong data. Tenants can share the same schema definitions, and the same application code. And there's no change at all to the application code. You just need to define a tenant in

the database for each distinct organization you serve, and then define users for each of those tenants.

Now it's time to get started setting up a real example. As I mentioned, in *my* example I'm assuming that I currently have multiple databases to support different customers. I want to combine them into a single new OpenEdge 11 database that's enabled for multi-tenancy:



There's another approach you might be taking, if you have used the technique I showed on the last few slides to combine multiple tenants in a single database in OpenEdge 10, with each identified by something that functions as a tenant id. In that case you want to be able to convert that one database to OpenEdge 11, get rid of the application-level tenant id dependency, and partition the data using the built-in multi-tenancy support in OpenEdge 11, as shown here:



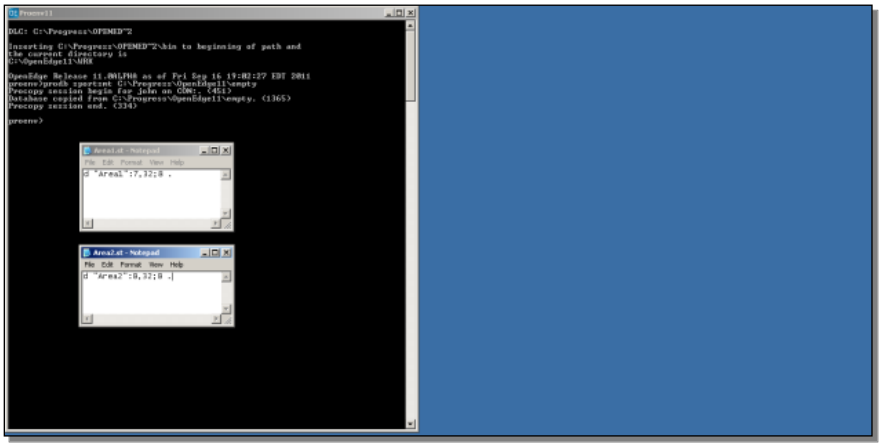
I will show an example of how to do that later in this series.

For now I'm preparing to load data from multiple databases into a single new database. In a **proenv** window, I can start by creating a new empty database called sportsmt.

```
proenv> prodb sportsmt empty
```

It's not my goal to explain the details of OpenEdge database internals, but you probably know that since OpenEdge 10 we have supported a new advanced storage

scheme called **Type II storage areas**. Suffice it to say for now that if you want to partition data for different tenants, you have to have one or more Type II storage areas defined for your database. The screenshot below shows two **.st** files that define a Type II storage area named **Area1**, with an internal area number of 7, 32 records per block, and a cluster size of 8; and a second .st file for **Area2**, whose internal area number is 8:



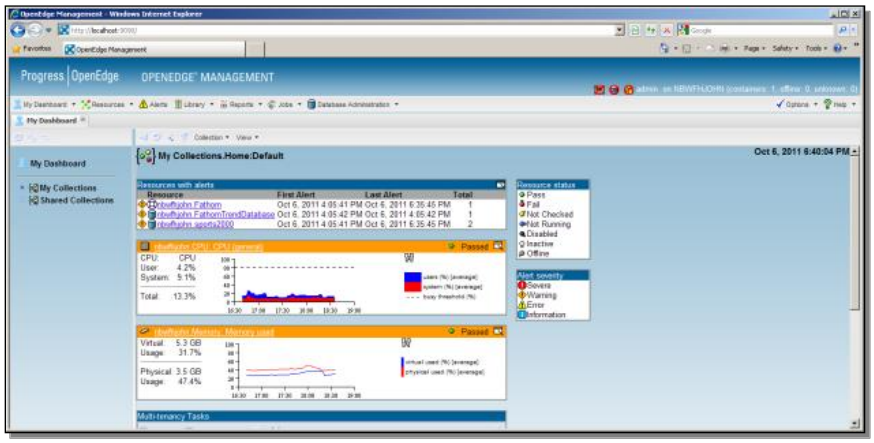
I run the **prostrct** utility to add Area1 to the database, and then Area2. With that I'm done setting up my new database.

```
proenv> prostrct add sportsmt Area1.st
proenv> prostrct add sportsmt Area2.st
```

Finally I start a server for the new database, so that I can access it from the supporting tools.

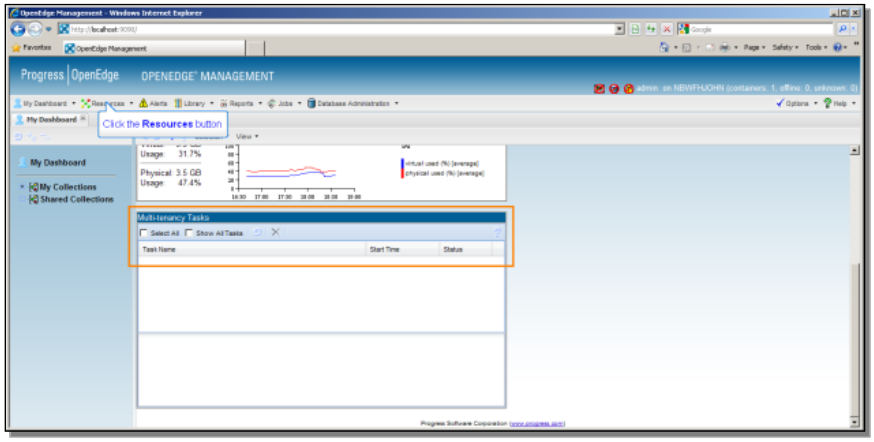
```
proenv> proserve sportsmt -H localhost -S 9100
```

Now it's time to start up the tool that I'll use to manage the new database:

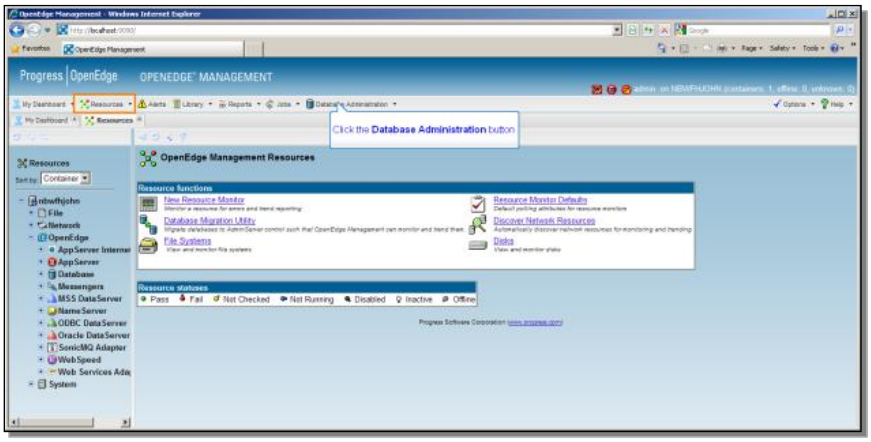


I'm showing **OpenEdge Management** here, which includes the management dashboard, but even if you're just running **OpenEdge Explorer**, you will also get a

dashboard, because there's a panel in it for monitoring the progress of multi-tenancy tasks that run in the background:

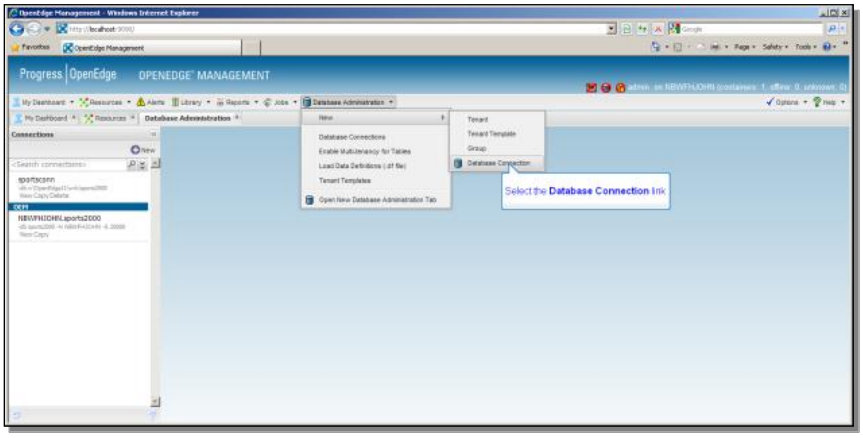


If I click the **Resources** tab, I see a treeview on the left, in what we call the list frame, that looks very much like the treeview you are familiar with from the OpenEdge 10 UI for OpenEdge Explorer. I can expand that, to look at the entries under OpenEdge, and see the same categories you're familiar with from OpenEdge 10:

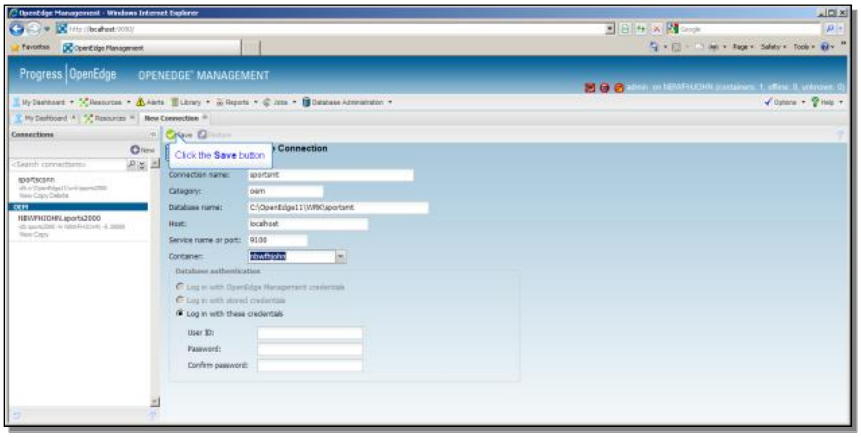


The new user interface provides a lot of valuable new capabilities. For instance, I could select the dropdown arrow next to Resources and create another new Resources tab, so I could manage different sets of settings in parallel, but I won't go into all that here. Instead I'll just click the **Database Administration** tab.

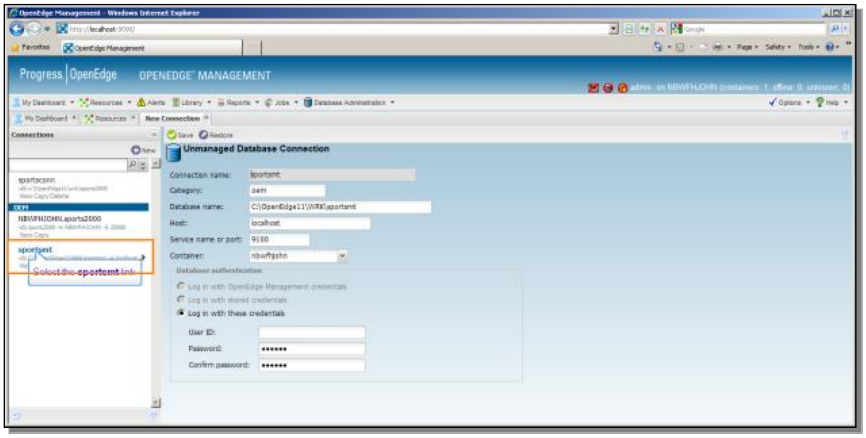
This brings up the **Database Administration Console**, which as I said is common to both OpenEdge Management and OpenEdge Explorer. Under its dropdown arrow, I can select **New**, and define a **new database connection**, for my sportsmt database server:



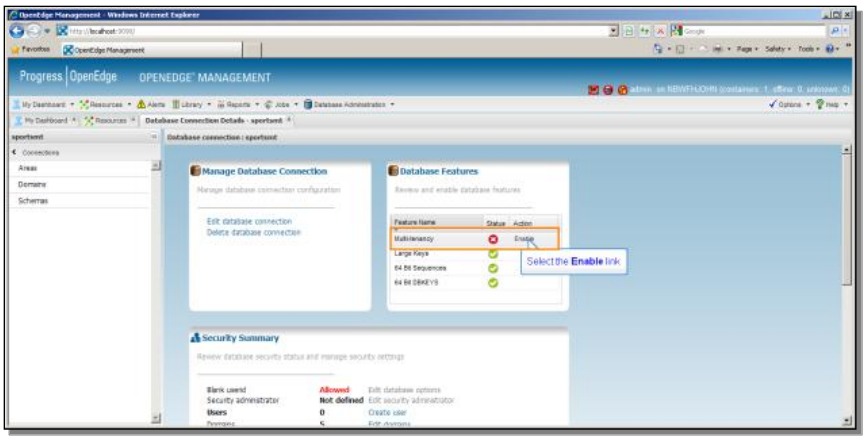
I make sportsmt the **connection name**, and then enter a **Category** which organizes all the connections in the list frame. The **OEM** category is for connections defined in OpenEdge Management or Explorer. Then I enter the pathname to the **database**, the **host**, and the **server** port number. The one **container** available to me is the name of my local machine where the database is running. I could also have remote containers as well. I won't specify anything for **login credentials**, so I'm done:



Below you can see the new database connection over in the list frame:



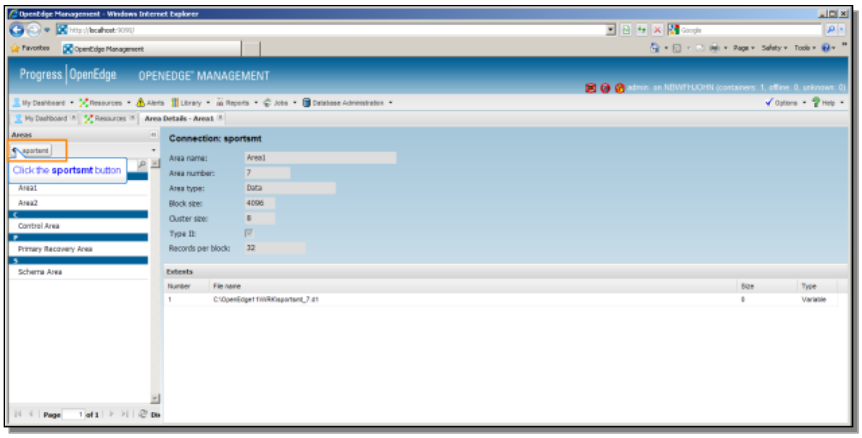
If I click on the database connection name, which is an active link, then a general display comes up in the details frame to the right showing various ways in which I can now manage the database through the tool. I make just one critical change here, and that's to enable the database for multi-tenancy:



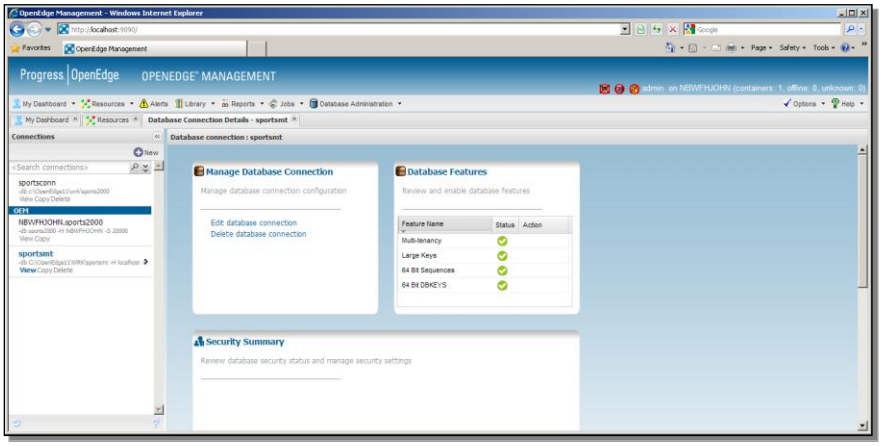
You can't enable an older database for multi-tenancy, of course, and you have to specifically enable an OpenEdge 11 database to support it. You also need to have an Enterprise database license to enable it. I confirm that I want to proceed, and multi-tenancy is now enabled for my sportsmt database.

Here are a couple of other things to show you in the DB Admin Console:

In the list frame, if I select the **Areas** link, I see my new Type II storage areas. Selecting one of them, I see the details of the Area definition from the .st file I loaded using prostrct:



Back in the list frame to the left, the '<' sign with the **sportsmt** label acts as a breadcrumb to let me walk backward through a series of displays I've been in, which is another useful new feature of the OpenEdge 11 user interface. From there I can go back to my **Connections** list. I can then select the **View** link under the sportsmt connection to see the connection information on the right, in the details frame, without changing the connections list on the left:



The first green checkmark indicates that multi-tenancy is now enabled for my new database, and I'm ready to start defining tenants and where their data goes. And that's what I'll do in the next episode in this series. In the meantime, you should know that for more detailed information there are two new manuals in the OpenEdge 11 documentation set devoted to multi-tenancy and how to manage it within OpenEdge Management and Explorer, one called **Getting Started with Multi-tenancy** and one on **Configuring Multi-tenancy**: