

Lesson 4 Transcript: DB2 Architecture

Slide 1: Cover

Welcome to Lesson 4 of the DB2 on campus series. Today we are going to talk about the DB2 architecture. My name is Raul Chong and I am the DB2 on Campus Program Manager.

Slide 2: Agenda

This is the agenda for today.

Slide 3: Agenda

We'll start with the "Process Model".

Slide 4: DB2 Process Model

This is the DB2 process model. It uses a threaded engine infrastructure. So, what you see here on the left side is a remote client application, and a local client application. They will go through a firewall. And then they will go through the DB2 engine. And **db2sysc** is the main DB2 process. And within the process, everything you see in circles and ellipses and are basically threads. Within **db2sysc**, there is a thread that is also called **db2sysc**, and that spawns the other threads as you can see this chart. There are different purposes for these different threads. For example, **db2ipccm** is used for a local client application connection. So, normally what will happen is that if a local client tries to connect to DB2, then this agent will be listening to the connection from the local client. And once this agent detects that connection, it will invoke an agent from the agent pool and then that agent will start working on behalf of this client application.

The same thing will happen for remote client application. But instead of **db2ipccm**, there will be different threads for different protocols. So, for example, if you are using TCP/IP, there will be an agent called **db2tcpcm** and that will be the one that will spawn or invoke another agent from this pool. And then that agent from that pool will start working with the remote client. So, an agent: you can think of an agent basically as a little worker that is doing things for DB2. Now, depending also on what you are doing on the database, there will be other threads that are spawned. For example, at the application level, there is a DB2 agent, which is the coordinator agent for all the workers and that agent may spawn other agents. For example, if you are using parallelism that we will cover later, and then it may spawn other agents to divide the work. Then at the "Database Level", you may have other threads that do different operations. For example, for log in, for pre-fetching, for dead-lock detection, for page cleaner, etc, etc. And at "Per Request", there are other types of threads. Now, on the left side you see other processes. So, again, anything you see as an ellipse or circle is a thread, and these ones are not threads. These ones are also processes.

Slide 5: Common Processes

So, these are the descriptions of the main DB2 processes. I already mentioned **db2sysc**, which is the main DB2 system controller. If you bring down that process then the entire DB2

system will crash. Obviously it will not be a good approach to bring down the system, because it may corrupt some things. It's basically bringing down, forcing DB2 down, not in a nice way. You have **db2acd**, the automatic computing daemon, the watch dog, which is the parent of the **db2sysc**, and so on and so on. You can pause this slide and you can read the description of each process for more detail.

Slide 6: Common Threads

Then we have threads. Again we have **db2sysc**. **db2sysc**, the name for that is for both, for the process, the main process, and within that process we have a thread that is called **db2sysc**. The thread is responsible for startup and shutdown the... that instance. We have also **db2tcpm**. I also mentioned this when I was explaining... just in the slide before, when a remote connection using TCP/IP tries to connect to the database. **db2agent**, etc, etc. Again, you can pause this slide... this presentation to take a look at this slide more carefully. Now, with DB2 9.5, or prior to DB2 9.5, these threads used to be processes in Linux and Unix.

In Windows, they were always threads. That was before 9.5. With 9.5, all of these used to be processes that were converted to threads. So basically, what I am trying to say here is, with DB2 9.5, on Linux, Unix and Windows we use exactly the same process model, where we have a few processes, and then the rest are threads. In the past, on Windows we used to have one main process and everything else was threads, and on Linux and Unix it used to be that most of these, or all of these were actually processes. So, by making the change in DB2 9.5, to be a more threaded architecture for the engine, we improved performance, and it allows us to simplify the configuration of, you know, many of the parameters of DB2.

Slide 7: Connection Concentrator

Ok, moving on to another topic, we have the "Connection Concentrator". What does this mean? In the past, for every connection, like in one of the applications, if you issued a **db2 connect** to the database sample, for example, then there would be an agent that would be allocated to handle that connection. If I have another connection to the same database, another agent will be created and so on and so on. So, if you have, let's say, 1000 connections, potentially you would need 1000 agents. And each agent uses a little amount of memory, let's say, from one to five megabytes of memory. So, you know, every connection implied more memory on the server for each agent. And, some companies just didn't have enough memory to handle this. So, connection concentrator was created to basically support that many connections could be handled by one agent. So, that's the purpose of having a Connection Concentrator, not a one-to-one relationship, that is, not one connection and one agent, but many connections and one agent. So, it's an n-to-m architecture.

Slide 8: Agenda

Ok, moving on to the next section of this presentation, we have the memory model.

Slide 9: DB2 Memory Model

And I'm just going to provide a very simple explanation of the memory model. We have different areas in memory. We have memory for the instance, or also known as the Database

Manager. We have memory at the database level, for the Database Global Memory and memory for applications and also for the agents. So, for example, when you issue a **db2start** for an instance, then this instance will be bringing up this area, the first rectangle that you see at the top that will be brought up from memory. So the *monitor heap* deducts the buffer size. So, basically this shows you that, when you start an instance, there's not much memory that is consumed. Now, for a database... for each database, you will have these areas of memory, and typically the buffer pool is one of the largest one for the database. Of course it's depending how you set the value of the buffer pool.

Now, you can have several buffer pools, so that can increase the memory. So, the first time you connect to a database... for the first time you activate a database, the memory that you see in this second rectangle, this one, would be brought up. So, the more databases are active, so many connections to different databases, then more memory will be used at the same time. And we have different areas in memory, the "Utility Heap", we have a "Backup Buffer", "Restore", "Package Cache". "Package Cache" basically is going to cache the access plan of different queries, so that if you repeat the same query, the access plan is already cached. We have the "Catalog Cache". That's basically, the catalog, or the data dictionary that DB2 uses very frequently. So this information can be cached in memory. Lock buffer for the locks, etc. etc. Then we have "Application Global Memory" for applications and "Agent Private Memory for agents, which, as I explained before when I was talking about DB2 process model, are the little workers that do things on behalf of DB2 on the server. And there are different areas that are going to be allocated on the "Application heap", the "Java heap", the "Sort heap", etc., etc.. Then we have also "Agent/Application Shared Memory" as well. Anyway, this is a very simple explanation of the memory model. You can search the DB2 developerWorks website for more articles about the DB2 memory model.

Slide 10: Agenda

Ok, now we are going to move and cover what is the Storage Model.

Slide 11: Buffer Pool Basics

So, we have a, first of all, what is called a buffer pool. And I talked about buffer pools when I was talking about the DB2 environment. A buffer pool is basically cache for tables and indices, caching memory. Memory is always going to help in reducing direct sequential I/O, so you don't have to access the disk all the time. You will be prefetching the information, so that means, you know, there may be cases where, either you don't need right now a given page, but there is this logic that would prefetch some pages in anticipation that you would need these pages in the future. Then you have the fact that buffer pools can be allocated in 4K, 8K, 16K and 32K pages. There should be at least one buffer pool per database, and there is a default one, which is IBMDEFAULTBP. That's the name of the default buffer pool. And the page size of the buffer pool must match the page size of the table space that is associated to this buffer pool.

Slide 12: Creating a Buffer Pool

What is the table space? To create a buffer pool, you can right click on the Control Center and

choose Create, or just use the **create bufferpool** command.

Slide 13: Create Buffer Pool Dialog

And this is the way a Create Buffer Pool dialog looks like. So you specify the buffer pool name, you specify for SAMPLE 16K, you specify the buffer pool page size. In this case it's 16K for the page size. The size of the buffer pool, what part is unblocked, and what part is blocked. So "Non-blocked" and "Blocked", what these are used for is sometimes you want the pages to be blocked contiguously to the buffer pool instead of in different places, and that allows for better performance for given operations. So "Blocked" is the area you want to be created for contiguous pages. And then you have some other choices here that you can try yourself later.

Slide 14: Table Space Basics

Then, as I was saying, in the lesson where we talked about DB2 environment, we talked about table spaces. And table spaces are, let me just bring up this chart that I used in that other lesson about DB2 environment.

Demo

A table space is basically what I've shown here as XYZ. You can think of it as a layer that is between your logical tables, Table A and Table B, and your physical Disk 1, Disk 2, and your physical memory, which this one here is a buffer pool. So basically if you want Table A to be stored in Disk 1, and use buffer pool BP, I first have to create a table space. In the syntax of the table space, I can specify where I want to store things of that table space, which disk, also what memory to use. And once I have created the table space, I can create a table and say *in* the table space. So you can say "create table A", I put the columns, and at the end I say *in tablespace "xyz"*. So that's how table spaces work.

Back to Slide 14

And, as I was saying, for table spaces you can specify containers, which are basically files and directories, or raw devices, where the actual data will be stored. And you can define... when you define a table space you can define the page size, you can define the extent size, which means how many pages... Basically it's not good for performance for individuals to just work with one page at a time. So what DB2 does is to use one extent at a time, where one extent is basically a bunch of pages. And in here, when you define table space, when you issue that **create tablespace** command, you can specify the "extent size"; "Pre-fetch size" is the same idea. How many pages you need to prefetch. So now I talked about that earlier, about bringing more pages than you need, in the anticipation that you will need them. Under "buffer pool", you have to specify which buffer pool is associated to this table space.

Slide 15: Table Space Types

There are four types, or three types, of table spaces: "REGULAR", for user tables, "LARGE", for large objects and also for XML, "TEMPORARY", we have two types within "TEMPORARY" table spaces. First we have "SYSTEM TEMPORARY", which... one example of this is TEMPSPACE1, which I explained when I was talking about the DB2

environment, for example, for sorts or for DB2 internal operations or for joins. So when there is not enough space in memory, then you may need to create tables within this SYSTEM TEMPORARY table space to finish its calculations. And once it has finished its calculations it will delete the tables that it temporarily created.

Then we have, within TEMPORARY table spaces, we have also "USER TEMPORARY" tables, which are used for declaring global temporary tables, which we briefly explained when we were covering the lesson about working with database objects. And USER TEMPORARY and SYSTEM TEMPORARY are often confused by users, you know, by common users. But you have to remember SYSTEM TEMPORARY table spaces are from the system. That means DB2 manages them, while USER TEMPORARY is managed by a user, a regular user of DB2 to explicitly create these temporary tables to work with... i.e., a temporary table space to work with temporary tables.

Slide 16: Table Space Management

Ok, now, in terms of table space management, we have "MANAGED BY SYSTEM", also known as SMS, which stands for System Managed Storage, where "System" in this case is a file system. It's fairly easy to manage. You don't need to pre-allocate anything, basically the file system will take care of this and the files will grow dynamically. And, there are some disadvantages of SMS. IF performance is very good, but it's not the best, you cannot split index and large objects to different table spaces. Everything has to be together.

With respect to the other management for table spaces, we have "MANAGED BY DATABASE", also known as "Database Managed Storage", or "DMS". Here you pre-allocate your files or raw devices. It allows you and gives you more flexibility in the sense you can add, drop and resize the containers, while in the SMS you could not do that. Oh, there is a way to do that, but it's just a little bit more complicated in a way. And then you could separate, in DMS, your index, your table and LOBs into separate table spaces.

Slide 17: Table Space Management

And starting with DB2 version 8.2, we started working on creating this other management type, which is called "MANAGEMENT BY AUTOMATIC STORAGE". There have been improvements in version 9 and version 9.5 where we were basically trying to get the best of both worlds, SMS and DMS; SMS in terms of ease of administration, and DMS also in terms of ease in terms of flexibility as to what you can do. So it is intended to be a "single point of storage management" for table spaces. What you need to do first is you create a database and you specify some storage path, which is basically where you want to store your data in your table spaces, and you don't explicitly need to define your containers. The containers will grow automatically in those directory paths, and they will be split, like cross directory paths. If the directory paths consist of different disks, information will be spread across all these disks.

Slide 18: Automatic Storage Examples

And, anyway, we can see here some examples of automatic storage. So, for example, the first is "CREATE DATABASE DB1 AUTOMATIC STORAGE YES". Actually this second line is

not needed because automatic storage is on by default. Then we have `CREATE DATABASE DB3 ON db path 1, db path 2`. That means we are creating this database DB3, which is created with automatic storage and we are also providing the storage paths. So these are the storage paths for this database. And then table spaces created within this database will be using these storage paths. Then we can say `CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE`.

Again, so we created a database, now we are creating a specific table space to use the automatic storage, because we can also just create it to use just managed by SMS, or managed by DMS. And then we can create a temporary table space as well. When we don't specify automatic storage, again, that will be the default. And we also can do, for example, `CREATE TABLESPACE TS1`. We don't to specify automatic storage, because that's default. But we can specify other things like `INITIALSIZE` that we want for the table space, `INCREASESIZE`, how much to increase. It can increase every now and then, and the maximum size (`MAXSIZE`), 100 M in this example.

Slide 19: Pages and Extents

Ok, so this slide shows how DB2 writes into disks. So, for example, we have three disks, and we have two tables: Table 1, which is shown with this color, or this pattern. And then we have here Table 2, which is shown with this other pattern. Now, these are extent. You cannot mix table information within extents. So, one extent is for one table. You cannot be mixing information of different tables in one extent. So, the way DB2 will start writing into these three disks is, it's going to write into the first disk, one extent, then to the other one, another extent to another disk and so on and so on, 3, 4, 5, and 6. And here we have another extent, but it is from Table 2, another extent from Table 2. And then we have Table 1 again, then it continues 9, 10, 11, etc., etc. And this is done by default. And by the way... the reason why DB2 writes information like this is for performance. We are basically spreading the data across different disks so that there will be less problems in terms of I/O, you know, accessing data, not everything. There will be no bottleneck on a given disk, or we are actually reducing that possibility because we are spreading information across different disks.

Slide 20: Creating a Table Space

To create a table space, if you want to do it from the Control Center, you can go to that Control Center, right-click on table spaces, and choose Create.

Slide 21: Create Table Space Wizard

And then, you will get this wizard. You will have to basically go through the information from the wizard to different tabs and provide information.

Slide 22: Agenda

Ok, now, the last section of this presentation is parallelism.

Slide 23: Query Parallelism

And for parallelism we have two types of parallelism. We have "Inter-Query Parallelism",

and "Intra-Query Parallelism". Inter-query parallelism is basically several queries are being performed on the database at the same time. Pretty much every database supports this. So you can have many database connections, and queries running against them. So this is basically inter-query parallelism. Then we have intra-query parallelism, and within intra-query parallelism we have two types, one is intra-partition, and the other one is inter-partition. Intra-partition is the one that is turned on using that parameter INTRA_PARALLEL, which is a dbm cfg parameter. So let's take a look at more details at this section, which is Intra-query Parallelism. And we are going to look at intra-partition parallelism and inter-partition parallelism.

Slide 24: Intra-partition Parallelism

So intra-partition... you know, if you kind of get this mixed up, just remember, Internet and intranet. You know, Internet is global, around the world, let's say, and intranet is more like within a company, for example. So, here we have "intra-partition", so it's within a partition in DB2. So let's say you have a "SELECT ... FROM ..." statement within a partition, that select statement could be parallelized to access the data. Ok, so that's intra-partition parallelism.

Slide 25: Inter-partition Parallelism

Then we have also inter-partition, which means across different partitions. You have a select statement, and you can parallelize that statement across different partitions, as you can see in this example. Now, this only works with the DPF option in DB2 Enterprise Edition. DPF stands for Database Partitioning Feature and in DB2 Express-C that feature is not provided.

Slide 26: Inter and Intra partition parallelism!

And also you can do both. You can have inter and intra partition parallelism. That means you have your SQL statement, and then... you have your SQL statement and within each of the partition. So, first you parallelize for different partitions and then within each partition you also have parallelism.

Slide 27: What's Next

Alright, so we have finished this lesson. So congratulations for completing this Lesson number 4 about DB2 architecture. And as to what is next, it will be Lesson 5 on client connectivity. So thank you very much for listening and have a good day.