Oracle 19c's Multitenant Container Architecture and Big Data

Sikha Bagui
https://orcid.org/0000-0002-1886-4582
University of West Florida, USA

Mark Rauch
University of West Florida, USA

INTRODUCTION

The importance of Big Data cannot be understated, and its value will continue to grow alongside it. With the exponential growth in data, developers have had to adjust database architecture in an to attempt to keep pace with Big Data. In late 2016 Oracle introduced a major architectural overhaul to its relational database system that would bring about changes needed to function in a market that supported Big Data. The introduction of multitenant architecture in 12c opened the door for Oracle and Oracle has continued to improve upon it. Recently, Oracle tagged its 19c version as a long-term release candidate for this architecture going forward. Historically Oracle has not been known for its ability to manage Big Data, but with these recent changes, it aims to remedy that. With Oracle 19c comes improvements in Oracle's container architecture.

The container architecture is a logical collection of data or metadata within a multitenant architecture, which allow for applications to be developed quickly and have workload portability. Hussein et al. (2019) discuss the idea of a container as a Service (CaaS) in a cloud environment. Multi-tenant architecture allows multiple users to share a single instance of a software application and its underlying resources. Containers, units within the container architecture, are bundled units of software with all their dependencies, packaged so that applications can run quickly and reliably when transported from one computing environment to another. Container architecture is being used by Google (Sanchez, 2014) and IBM (IBM Cloud, n.d.). Google, who has been using container architecture since 2004, is integrating containers into it's Cloud Platform (Sanchez, 2014). Red Hat-IBM has been named the leader in multi-cloud container development ("Containers on IBM Cloud", n.d.).

Oracle 19c's new multitenant architecture enables an Oracle database to function as a multitenant container database (CDB). A CDB is composed of pluggable databases (PDBs), which are a movable collection of schemas, schema objects and non-schema objects ("Administrator's Guide," n.d.).

In this chapter, the architectural changes that come with 19c multitenant architecture are presented. The architectural features discussed are: automated indexing, the online table move feature, pluggable database relocation, rapid home provisioning, preventing schema-only locks, and auditing. These are presented in the form of case studies, demonstrating the improvements that come with 19c when compared with an environment that is currently running Oracle 11g.

The rest of this chapter is organized as follows. Section two presents the traditional oracle architecture; section three presents the multitenant architecture, including the advantages of the multitenant architecture and shared multitenant architecture; section four presents big data and oracle in the context of big

DOI: 10.4018/978-1-7998-9220-5.ch031

data, including how multitenant architecture resolves issues that were a problem in the past; section five describes the experimentation, the grid and database installs; section six demonstrates the improvements of the multitenant architectural features in Oracle 19c; and section seven presents the conclusions.

BACKGROUND: TRADITIONAL ORACLE ARCHITECTURE

Oracle databases traditionally consist of at least one database instance. The database itself is a set of files that store its data, and these files can exist independently of the database instance. The database instance is a set of memory structures called the system global area (SGA) and a set of background processes. The instance can exist independently of the database files. When a user connects to the Oracle database, a client process is created with its own server process. Each server process has its own private session memory known as program global area (PGA) (¨Oracle® Database. Database Concepts," 2021). It is through the application that interactions are made across either multiple logical databases within a single physical database, or a single logical database distributed across multiple physical databases (¨Oracle® Database. Database Concepts," 2021). Each of these memory structures have far more depth to them inclusive of the many different processes hosted within. This traditional structure has been deprecated in favor of a new one, hence the focus of this chapter.

FOCUS OF ARTICLE: MULTITENANT ARCHITECTURE

Buyya et al. (2018) and Hillman et al. (2021) have looked at multi-tenant architecture. They have discussed multi-tenant architecture from a scheduling perspective. Oracle 19c uses a multitenant architecture, which enables an Oracle database to be a CDB. A container database can contain either zero, one, or many pluggable databases (PDB). A PDB is a portable collection of schemas, schema objects, and non-schema objects that appears to an Oracle Net client as a non-CDB ("Oracle® Database. Database Concepts," 2021). This portable and pluggable database has many benefits and it solves several problems that have plagued the traditional architecture.

Advantages of the Multitenant Architecture

Whether large or small, there are benefits to migrating to this new architecture, though the larger enterprises will see more merit. Large enterprises may use hundreds or thousands of databases, often running on multiple physical servers. Current technology allows for this and can withstand heavy workloads. Historically a database may only use a fraction of that server's hardware capacity, and this would lead to either under provisioning equipment or over provisioning it. Multitenant architecture allows the consolidation of multiple physical databases across different hardware into a single database structure on a single piece of hardware. Broadly speaking, the immediate benefits of using a multitenant architecture include (¨Oracle® Database. Database Concepts," 2021):

- Hardware cost reduction
- More efficient movement of code
- Easier management and monitoring of the physical database
- Separation of raw data and code