

**OCEANBASE**



# OceanBase 数据库

## OceanBase 术语

| 产品版本：V4.3.5


| 文档版本：20250106

# 声明

**北京奥星贝斯科技有限公司版权所有©2024，并保留一切权利。**

未经北京奥星贝斯科技有限公司事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

## 商标声明

 **OCEANBASE** 及其他 OceanBase 相关的商标均为北京奥星贝斯科技有限公司所有。本文档涉及的第三方的注册商标，依法由权利人所有。

## 免责声明

由于产品版本升级、调整或其他原因，本文档内容有可能变更。北京奥星贝斯科技有限公司保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在北京奥星贝斯科技有限公司授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过北京奥星贝斯科技有限公司授权渠道下载、获取最新版的用户文档。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

# 通用约定

格式	说明	样例
 <b>危险</b>	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 <b>危险</b> 重置操作将丢失用户配置数据。
 <b>警告</b>	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	 <b>警告</b> 重启操作将导致业务中断，恢复业务时间约十分钟。
 <b>注意</b>	用于警示信息、补充说明等，是用户必须了解的内容。	 <b>注意</b> 权重设置为0，该服务器不会再接受新请求。
 <b>说明</b>	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 <b>说明</b> 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击 <b>设置&gt; 网络&gt; 设置网络类型</b> 。
<b>粗体</b>	表示按键、菜单、页面名称等UI元素。	在 <b>结果确认</b> 页面，单击 <b>确定</b> 。
Courier字体	命令或代码。	执行 <code>cd /d C:/window</code> 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	<code>bae log list --instanceid</code> <code>Instance_ID</code>
[ ] 或者 [a b]	表示可选项，至多选择一个。	<code>ipconfig [-all -t]</code>
{ } 或者 {a b}	表示必选项，至多选择一个。	<code>switch {active stand}</code>

# 目录

1 OceanBase 术语	13
1.1 A	13
1.1.1 ACID	13
1.1.2 Antman (简称 oat-cli)	13
1.1.3 ASH (Active Session History)	13
1.1.4 ASH 报告 (OceanBase Active Session History Report)	13
1.2 B	13
1.2.1 保护模式 (Protection Mode)	13
1.2.2 备份元数据库/恢复元数据库 (Backup Metadb/Restore Metadb)	13
1.2.3 备份工具程序/恢复工具程序 (AgentServer/agentrestore.jar)	14
1.2.4 本地事务 (Local Transaction)	14
1.2.5 本地计划 (Local Plan)	14
1.2.6 本地执行 (Local Execution)	14
1.2.7 表 (Table)	14
1.2.8 表空间 (Tablespace)	14
1.2.9 表组 (Table Group)	14
1.2.10 并行查询 (Parallel Query)	15
1.2.11 并行合并 (Parallel Compaction)	15
1.2.12 Block Cache	15
1.2.13 BloomFilter Cache	15
1.2.14 并行执行 (Parallel Execution)	15
1.2.15 布隆过滤器 (Bloom Filter)	15
1.2.16 版本升级	15
1.3 C	16
1.3.1 Clog (Commit Log)	16
1.3.2 CDC (Change Data Capture, 即变更数据捕获)	16
1.3.3 磁盘静默错误 (Silent Data Corruption)	16
1.3.4 查询改写 (Query Rewrite)	16
1.3.5 存储过程 (Stored Procedure)	16

1.4 D	16
1.4.1 Data Flow Object (DFO)	16
1.4.2 Data Transfer Layer (DTL)	16
1.4.3 DBaaS (Database as a service)	17
1.4.4 DIO (Direct Input-output)	17
1.4.5 单机版 (Standalone Edition)	17
1.4.6 DOOBA	17
1.4.7 大版本冻结 (Major Freeze)	17
1.4.8 大版本冻结版本号 (Major Freeze Version)	17
1.4.9 冻结版本 (Frozen Version)	17
1.4.10 地域 (Region)	17
1.4.11 地域/互联数据中心 (Region/IDC)	17
1.4.12 冻结 MemTable (Frozen MemTable)	18
1.4.13 读写 Zone/只读 Zone	18
1.4.14 对象存储 (Object Storage)	18
1.4.15 多级缓存 (Multi-level Cache)	18
1.4.16 多租户 (Multi-tenant)	18
1.5 E	18
1.5.1 二级索引 (Secondary Index)	18
1.6 F	19
1.6.1 反向增量 (Reverse Incremental)	19
1.6.2 访问路径 (Access Path)	19
1.6.3 分布式事务 (Distributed Transaction)	19
1.6.4 分布式计划 (Distributed Plan)	19
1.6.5 分布式执行 (Distributed Execution)	19
1.6.6 分片 (Tablet)	19
1.6.7 分区 (Partition)	19
1.6.8 分区表 (Patition Table)	20
1.6.9 分区裁剪 (Partition Pruning)	20
1.6.10 分区键 (Partition Key)	20
1.6.11 分区副本 (Partition replica)	20
1.7 分区转移 (Transfer)	20
1.8 分页保序	20
1.8.1 副本 (Replica (V2.x/V3.x 版本))	20
1.8.2 副本 (Replica (V4.x 版本))	22
1.8.2.1 注意	23
1.8.3 负载均衡 (Load Balance)	24
1.8.4 复制 (Replication)	25
1.8.5 复制表 (Replicated Table)	25
1.8.6 Fuse Row Cache	25

1.9	G	25
1.9.1	高可用 (High Availability)	25
1.9.2	Granule	25
1.9.3	工作线程 (Worker Threads)	25
1.9.4	归一化 (Normalization)	25
1.9.5	故障切换 (Failover)	26
1.10	H	26
1.10.1	Hint	26
1.10.2	行 (Row)	26
1.10.3	行版本合并 (Row Compact)	26
1.10.4	行缓存 (Row Cache)	26
1.10.5	合并/每日合并 (Major Compaction)	26
1.10.6	宏块 (Macro Block)	26
1.10.7	宏块分裂 (Macro Block Split)	27
1.10.8	宏块合并 (Macro Block Merge)	27
1.10.9	宏块空间回收 (Macro Block Recycle)	27
1.10.10	宏块预读 (Macro Block Prefetch)	27
1.10.11	活跃 MemTable (Active MemTable)	27
1.10.12	活跃事务 (Active Transaction)	27
1.11	J	27
1.11.1	集群 (Cluster)	27
1.11.2	集群实例 (Cluster Edition)	27
1.11.3	基表 (Base Table)	28
1.11.4	基线数据 (Baseline Data)	28
1.11.5	基线数据版本 (Baseline Data Version)	28
1.11.6	级联复制 (Cascading)	28
1.11.7	渐进合并 (Progressive Compaction)	28
1.11.8	监控系统 (Monitoring System)	28
1.11.9	OBServer 节点, 节点 (OBServer Node, Node)	28
1.12	K	28
1.12.1	可用区/区 (Availability Zone, AZ)	28
1.12.2	库, 数据库 (Database)	29
1.12.3	快速参数化 (Fast Parsing)	29

1.13 L	29
1.13.1 Label Security	29
1.13.2 Learner	29
1.13.3 Liboblog	29
1.13.4 联接顺序 (Join Order)	29
1.13.5 联接算法 (Join Algorithm)	29
1.13.6 链路 (Pipeline)	30
1.13.7 Location Cache	30
1.13.8 Log Group	30
1.13.9 轮转合并 (Rotating Major Compaction)	30
1.13.10 逻辑数据中心 (Logical Data Center, LDC)	30
1.14 M	30
1.14.1 MaaS (Management as a service)	30
1.14.2 Meta 租户 (Meta Tenant)	30
1.14.3 MVCC (Multi-version Concurrency Control)	31
1.14.4 慢查询 (Slow Query)	31
1.14.5 Membership Log	31
1.14.6 Mini SSTable	31
1.14.7 Minor SSTable	31
1.14.8 Major SSTable	31
1.14.9 Multi-Paxos	31
1.14.10 模糊绑定 OUTLINE	32
1.14.11 MySQL 模式 (MySQL Mode)	32
1.14.12 mysqltest	32
1.15 N	32
1.15.1 Nop Log	32

1.16 O	32
1.16.1 OBKV-Table	32
1.16.2 OBKV-HBase	32
1.16.3 OBAgent	33
1.16.4 ob_admin	33
1.16.5 ob_error	33
1.16.6 observer 进程 (Observer Process)	33
1.16.7 OBServer 服务器 (OBServer Host)	33
1.16.8 OceanBase 安装部署工具 (OceanBase Deployer)	33
1.16.9 OceanBase C API 库 (OceanBase Connector/C)	33
1.16.10 OceanBase Cloud	34
1.16.11 OceanBase 导数工具 (OceanBase Loader and Dumper)	34
1.16.12 OceanBase 管理者工具 (OceanBase Admin Toolkit, OAT)	34
1.16.13 OceanBase JDBC 驱动程序 (OceanBase Connector/J)	34
1.16.14 OceanBase 开发者中心 (OceanBase Developer Center, ODC)	
1.16.15 OceanBase 命令行客户端 (OceanBase Command-Line Client, OBClient)	34
1.16.16 OceanBase OCI 驱动程序 (OceanBase Call Interface, OBCI)	35
1.16.17 OceanBase ODBC 驱动程序 (OceanBase Connector/ODBC)	35
1.16.18 OceanBase 嵌入式 SQL 预编译器 (Embedded SQL in C for OceanBase, ECOB)	35
1.16.19 OceanBase 迁移服务 (OceanBase Migration Service, OMS)	35
1.16.20 OceanBase 迁移评估 (OceanBase Migration Assessment, OMA)	
1.16.21 OceanBase 数据库 (OceanBase Database)	36 36
1.16.22 OceanBase 数据库企业版 (OceanBase Database Enterprise Edition)	36
1.16.23 OceanBase 数据库社区版 (OceanBase Database Community Edition)	36
1.16.24 OceanBase 数据库代理 (OceanBase Database Proxy, ODP)	36
1.16.25 OceanBase 云平台 (OceanBase Cloud Platform, OCP)	36
1.16.26 OceanBase 云平台 Express (OCP Express)	37
1.16.27 OLAP (On-line Analytical Processing)	37
1.16.28 OLTP (On-line Transaction Processing)	37
1.16.29 Oracle 模式 (Oracle Mode)	37
1.17 P	37
1.17.1 Partition 调度 (Partition Scheduling)	37
1.17.2 Piece	38
1.17.3 PX worker	38



1.18 Q	38
1.18.1 Query Coordinator (QC)	38
1.18.2 迁入 (Move In)	38
1.18.3 迁出 (Move Out)	38
1.18.4 迁移 (Migration)	38
1.18.5 全链路追踪 (End to End Tracing)	38
1.18.6 全量合并 (Full Compaction)	38
1.18.7 全量校验 (Full Verification)	39
1.18.8 全量刷新 (Complete Refresh)	39
1.18.9 全量数据导入 (Initial Load)	39
1.18.10 全量提取 (Initial Load Extract)	39
1.18.11 全局时间戳服务 (Global Timestamp Service, 简称 GTS)	39
1.18.12 全局索引 (Global Index)	39
1.18.13 全局一致性快照 (Global Consistent Snapshot)	39
1.18.14 强一致性读/弱一致性读 (Strong Read Consistency/Weak Read Consistency)	40
1.19 R	40
1.19.1 Redo 日志索引 (Redo Log Index)	40
1.19.2 Rebuild	40
1.19.3 日志服务 (Log Service)	40
1.19.4 日志归档 (Log Archive)	40
1.19.5 日志流 (Log Stream, LS)	40
1.19.6 日志流组 (Log Stream Group, LS Group)	41
1.19.7 容器服务 (Container Service)	41
1.19.8 容错 (Fault Tolerance)	41
1.19.9 Root Service (RS)	41
1.19.10 Rowid	41
1.19.11 Row Merge	41
1.19.12 RPC (Remote Process Call)	41
1.19.13 RS List	42
1.19.14 Reconfirm	42

1.20 S	42
1.20.1 SaaS (Software-as-a-service)	42
1.20.2 Savepoint	42
1.20.3 Schema	42
1.20.4 Schema Version	42
1.20.5 闪回查询 (Flashback)	42
1.20.6 视图 (View)	43
1.20.7 数据编码 (Encoding)	43
1.20.8 数据订阅 (Data Subscription)	43
1.20.9 数据复制 (Data Replication)	43
1.20.10 数据获取 (Data Ingestion)	43
1.20.11 数据集成 (Data Integration)	43
1.20.12 数据均衡 (Data Balancing)	43
1.20.13 数据迁移 (Data Migration)	44
1.20.14 数据转换 (Data Transformation)	44
1.20.15 Slog/SSTable Log	44
1.20.16 Server List	44
1.20.17 sql_diagnoser	44
1.20.18 SQL Plan Management (SPM)	44
1.20.19 Sorted Strings Table (SSTable)	44
1.20.20 Sub Query Coordinator (SQC)	44
1.20.21 数据库 (Database)	44
1.20.22 数据倾斜 (Data Skew)	45
1.20.23 刷新 Schema (Schema Refresh)	45
1.20.24 算子 (Operator)	45
1.21 T	45
1.21.1 统计信息 (Statistics)	45
1.21.2 透传模式 (Pass-through Mode)	45
1.21.3 TPC-C	45
1.21.4 TPC-H	45
1.22 U	45
1.22.1 Unit 调度 (Unit Scheduling)	46
1.22.2 Unit Number	46
1.22.3 Unit Group	46
1.22.4 Universe	46

1.23 W	46
1.23.1 微块 (Micro Block)	46
1.23.2 微块缓存 (Block Cache)	47
1.23.3 微块索引缓存 (Block Index Cache)	47
1.23.4 位置 (Locality)	47
1.23.5 位置信息缓存 (Location Cache)	47
1.23.6 物化视图 (Materialized View)	47
1.23.7 物化视图日志 (Materialized View Log)	47
1.23.8 无损切换 (Switchover)	48
1.23.9 无主选举 (Election without leader)	48
1.24 X	48
1.24.1 XA 事务 (XA Transaction)	48
1.24.2 系统变量 (System Variables)	48
1.24.3 系统配置项 (System Parameters)	48
1.24.4 系统租户 (sys Tenant)	48
1.24.5 小版本冻结 (Minor Freeze)	48
1.24.6 虚拟交换机 (vSwitch)	49
1.25 Y	49
1.25.1 延时 (Latency)	49
1.25.2 业务连续性 (Business Continuity)	49
1.25.3 异地多活 (Active Geo-redundancy)	49
1.25.4 异地灾备 (Geo-redundancy)	49
1.25.5 用户租户 (User Tenant)	49
1.25.6 远程计划 (Remote Plan)	49
1.25.7 远程执行 (Remote Execution)	50
1.25.8 源端 (Source Connection Profile)	50
1.25.9 优化器 (Optimizer)	50
1.25.10 有主改选 (Re-election with leader)	50
1.25.11 预检查 (Pre-Check)	50
1.25.12 云服务器 (Cloud Server)	50

1.26 Z	50
1.26.1 Zone	50
1.26.2 增量合并 (Incremental Compaction)	51
1.26.3 增量数据 (Incremental Data)	51
1.26.4 增量数据 (Change Data)	51
1.26.5 增量数据表 (MemTable)	51
1.26.6 增量同步 (Change Synchronization)	51
1.26.7 只读可用区 (Read Zone)	51
1.26.8 正向切换 (Forward Shifting)	51
1.26.9 执行计划/计划 (Execution Plan/Plan)	51
1.26.10 执行计划缓存/计划缓存 (Plan Cache)	51
1.26.11 执行计划绑定 (Execution Plan Binding)	52
1.26.12 执行计划匹配 (Execution Plan Matching)	52
1.26.13 仲裁服务 (Arbitration Service)	52
1.26.14 仲裁成员 (Arbitration Member)	52
1.26.15 物理备库 (Physical Standby Database)	52
1.26.16 主/从副本 (Leader/Follower)	52
1.26.17 主租户 (Primary Tenant) 和备租户 (Standby Tenant)	53
1.26.18 主键 (Rowkey)	53
1.26.19 主可用区 (Primary Zone)	53
1.26.20 转储 (Mini Compaction Minor Compaction)	53
1.26.21 专有云网络 (Virtual Private Cloud)	53
1.26.22 子计划 (Sub Plan)	53
1.26.23 自适应游标共享 (Adaptive Cursor Sharing)	53
1.26.24 资源池 (Resource Pool)	54
1.26.25 资源单元 (Unit)	54
1.26.26 资源管理计划 (Resource Management Plan)	54
1.26.27 资源管理计划内容 (Resource Management Plan Config)	54
1.26.28 资源规格 (Unit Config)	54
1.26.29 资源组 (Resource Group)	54
1.26.30 租户 (Tenant)	54
1.26.31 租户实例 (Tenant Instance)	55
1.26.32 组提交 (Group Commit)	55
1.26.33 最大性能 (Maximum Performance)	55
1.26.34 最小化停机迁移 (Minimal Downtime Migration)	55
1.26.35 仲裁服务	55
1.26.36 仲裁 Server	55
1.26.37 仲裁成员	55
1.26.38 仲裁副本	55
1.26.39 仲裁降级	55
1.26.40 仲裁升级	56

# 1 OceanBase 术语

## 1.1 A

### ACID

数据库在进行数据更新时，为保证数据的可靠和一致，必须具备的 4 个特性：原子性（atomicity）、一致性（consistency）、隔离性（isolation）、持久性（durability）。

### Antman（简称 oat-cli）

提供一键部署 OCP、OMS、ODC 等 OceanBase 数据库周边工具平台的 CLI 工具。

### ASH（Active Session History）

一种活动会话历史记录的诊断工具，用于记录数据库中所有活动会话的信息。

### ASH 报告（OceanBase Active Session History Report）

能够提供定位瞬时发生异常的分析报告，与性能报告相比，能提供更加细粒度的诊断信息。一般的性能报告所覆盖的是小时级别的快照信息，诊断问题的粒度不能深入到 Session 级别，导致一些瞬时抖动信息很难从性能报告上得到详细的执行细节。因此，我们可以通过 ASH 报告这样一个会话级别的细粒度诊断信息来解决这种问题。

## 1.2 B

### 保护模式（Protection Mode）

OceanBase 数据库的物理备库仅支持最大性能模式。关于最大性能模式的详细介绍，参见本文中的 **最大性能（Maximum Performance）**。

### 备份元数据库/恢复元数据库（Backup Metadb /Restore Metadb）

备份元数据库包含参数表 `backup_base_profile` 以及控制备份任务的四张表，分别是 `base_data_backup`、`base_data_backup_task`、`base_data_backup_task_history` 和 `inc_data_backup`。恢复元数据库包含控制恢复任务的四张表，分别是 `oceanbase_restore`、`base_data_restore`、`inc_data_restore` 和

`oceanbase_restore_history`。通常会将备份元数据库和恢复元数据库部署在同一个数据库中。

该术语在 V4.x 版本废弃。

## 备份工具程序/恢复工具程序（AgentServer /agentrestore.jar）

AgentServer 是备份工具，是一个常驻进程，每隔一段时间查询元数据库 MetaDB 中的 `base_data_backup` 表有无备份任务，来控制整个基线、增量数据备份的发起、取消，也会随着任务的推进更新备份任务四张表的状态。`agentrestore.jar` 是恢复工具，顾名思义是 Java 编写的 `jar` 包，也是常驻进程，每隔一段时间查询元数据库 MetaDB 中的控制表，负责调度整个恢复任务的发起，也会随着任务的推进更新恢复任务四张表的状态。

该术语在 V4.x 版本废弃。

## 本地事务（Local Transaction）

是相对于跨机分布式事务而言的，特指事务所操作的表的日志流 `Leader` 全部在同一个 Server 上，并且与 Session 建立的 Server 具有相同的事务。

## 本地计划（Local Plan）

当执行计划只涉及到单表或分区表的单个分区，且该表或分区在本节点时，该计划为"本地计划"。

## 本地执行（Local Execution）

接收客户端请求生成执行计划的数据库服务器和计划实际执行的服务器是同一个。

## 表（Table）

最基本的数据库对象。每个表由若干行记录组成，每一行有相同的预先定义的列。用户通过 SQL 语句对表进行增、删、查、改等操作。通常，表的若干列会组成一个主键，主键在整个表的数据集合内唯一。

## 表空间（Tablespace）

OceanBase 数据库的加密在形式上尽量兼容 Oracle 数据库，数据的加密单位为表空间（Tablespace）。OceanBase 数据库并非一个多数据文件的数据库系统，表空间是为了兼容而设计的概念，可以简单理解为表空间是一组表的集合。

## 表组（Table Group）

对经常会被同时访问的一组表，为了优化性能，需要将它们相同类型的副本存储在同一个 OceanBase 数据库服务器中。通过定义一个 Table Group，并且将这一组表放在这个 Table Group 中来达到这个目的。此外，同一个 Table Group 的多个分区表具有相同的分区数和分区规则。假设一个 Table Group 里的表都有 N 个分区，所有这些表的第 i 个分区的集合组成一个 Partition Group。同一个 Partition Group 里的分区，主副本总是位于同一个 server 上。

## 并行查询（Parallel Query）

并行查询是指通过对查询计划的改造，提升对每一个查询计划的 CPU 和 IO 处理能力，从而缩短单个查询的响应时间。并行查询技术可以用于分布式执行计划，也可以用于本地查询计划。

## 并行合并（Parallel Compaction）

对于不同的数据分区，合并是会并行来做的。但是由于数据倾斜，某些分区的数据量可能非常大。尽管增量合并极大减少了合并的数据量，对于一些更新频繁的业务，合并的数据量仍然非常大，为此 OceanBase 数据库引入了分区内并行合并。合并会将数据划分到不同线程中并行做合并，极大地提升了合并速度。

## Block Cache

类似于其他数据库系统的 Buffer Cache，缓存具体的数据微块，每个微块都会解压后装载到 Block Cache 中，因此每个 cache 大小都是变长的。

## BloomFilter Cache

OceanBase 数据库的 BloomFilter 是构建在宏块上的，根据用户实际空查率按需自动构建，当一个宏块上的空查次数超过某个阈值时，就会自动构建 BloomFilter，并将 BloomFilter 放入 Cache。

## 并行执行（Parallel Execution）

一个执行计划在执行前，根据需要访问的分区情况，被切分成一个或者多个任务来执行。在执行过程中，调度器可以串行调度计划的多个任务，也可以让多个任务同时被调度执行。这种多个任务被同时调度执行的方式称为并行执行。

## 布隆过滤器（Bloom Filter）

布隆过滤器，用于快速判断行在基线数据或转储数据是否存在，当结果为不存在时，可以减少磁盘 IO 和 CPU 消耗。

## 版本升级

指的是 OceanBase 数据库软件版本升级，需要与“仲裁升级”的概念区分开。



## 1.3 C

### Clog (Commit Log)

OceanBase 会将包括事务在内的所有对数据库状态的修改操作，以 redo log 的形式记录并通过 Multi-Paxos 一致性协议持久化在集群的多个副本之中。这样的一组 redo log 在 OceanBase 中称作 clog。

Clog 按照日志流进行组织，每个日志流有自己的一组 clog 文件。同时，各个租户的 clog 文件是隔离的，每个租户可以设置本租户可以使用的 clog 空间大小。

Clog 既用于在数据库实例故障的场景下保证数据库的持久性和原子性，同时也用于备节点、物理备库、Change Data Capture 等多个数据库功能实时获取数据库的状态变更操作。

### CDC (Change Data Capture, 即变更数据捕获)

CDC 是一种用于捕获和记录数据库中数据变更的技术。当数据库中的数据发生变化时（例如插入、更新或删除数据），CDC 技术会按照变更发生的顺序捕获变更并记录下来，以便其他系统可以及时地处理和使用这些最新的数据。

### 磁盘静默错误 (Silent Data Corruption)

是指存储系统向应用程序提供损坏的数据，而未发出任何警告。例如由于介质损坏，磁盘中出现了坏块，在应用程序读取该块的时候，就会读到错误的数据。

### 查询改写 (Query Rewrite)

通过对用户查询做等价的改写以便优化器生成最佳执行计划的过程。

### 存储过程 (Stored Procedure)

服务器端提供的用户编程方式。

## 1.4 D

### Data Flow Object (DFO)

并行计划以数据重分布点为边界，切分为可以并行执行的逻辑子计划，每个子计划由一个 DFO 进行封装。

### Data Transfer Layer (DTL)

数据传输层，分布式并行执行框架中用于提供各执行线程之间数据传输的网络传输框架。



## DBaaS (Database as a service)

数据库即服务，是指客户直接通过 SaaS 服务模式来采购方式采购数据库云服务。数据库云服务商拥有所有的资源，包括底层的 IAAS，数据库软件，客户对于底层资源不感知，由数据库云服务商负责运维整个数据库产品，其实 DBaaS 是 SaaS 服务的一种。

## DIO (Direct Input-output)

绕过操作系统 page cache 的读写文件方法。

## 单机版 (Standalone Edition)

OceanBase 数据库的单机分布式一体化架构的数据库形态。

## DOOBA

OceanBase 数据库内部的一个运维脚本，用于性能监控，使用 Python 语言开发，且仅支持 Python 2.7。DOOBA 的原理是使用 MySQL 命令连接到 OceanBase 的 SYS 租户中，实时展示租户 SQL 的 QPS（包括 select、update、insert、delete、commit）以及相应 SQL 的平均延时（RT）。同时还可以展示各个节点的 SQL 的 QPS 以及 RT 等。

## 大版本冻结 (Major Freeze)

集群中所有的节点在一个统一的快照点冻结当前的活跃 MemTable，不再接受新开启事务的写操作，新事务的写操作在新的活跃 MemTable 中进行。

## 大版本冻结版本号 (Major Freeze Version)

大版本冻结的版本号。

## 冻结版本 (Frozen Version)

发生冻结操作时的版本号。

## 地域 (Region)

Region 是指一个地域或城市，而一个 Region 可能包含一个或多个 Zone。不同 Region 通常距离较远，而在 OceanBase 数据库中，一份数据可以被部署在多个 Region 中的不同副本上，以保证数据的高可用性和容错性。每个 Region 都是一个物理数据中心的地理区域，不同地域之间是完全隔离的，以确保系统在不同地域间具备最高程度的稳定性和容错能力。

## 地域/互联数据中心 (Region/IDC)

每台 OBCS 服务器都具有 region/IDC 属性，其中 region 记录 OceanBase 集群地域信息，通常代表一个城市，IDC 记录 OceanBase 集群的机房信息。一个 OceanBase 集群包含若干个 region，每个 region 包含若干个 IDC，每个 IDC 部署若干个 OBCS 服务器。根据不同 region 和 IDC，OceanBase 客户端与 OBCS 服务器，或者 OBCS 服务器 OBCS 服务器之间的位置关系可以分为 3 种：同 region 同 IDC、同 region 不同 IDC 和不同 region。三者优先级依次降低。

## 冻结 MemTable (Frozen MemTable)

Active MemTable 达到一定的内存阈值，进行冻结生成冻结 MemTable，冻结的 MemTable 不能再写入增量数据。

## 读写 Zone/只读 Zone

传统的读写分离架构中，写库与读库是完全不同的数据库实体，它们天然隔离的，通过数据同步工具实时同步。OceanBase 数据库通过只读副本和只读 Zone 的功能，实现了单个数据库实体下的读写分离支持。其中，读写 Zone 可以接受任何读写请求，只读 Zone 只能接受"登录认证/弱一致性读请求/select 不指定表名的请求 /use database/set session variables 语句"。

从 V2.2.3 版本不再支持只读 Zone 功能。

## 对象存储 (Object Storage)

用于存储非结构化数据的数据存储架构，它将数据划分为单元（对象），并存储在结构扁平的数据环境中。每个对象都包含数据以及应用可用于轻松访问和检索对象的元数据和唯一标识符。对象存储解决方案非常适合用于构建需要扩展和灵活性的云原生应用程序。

## 多级缓存 (Multi-level Cache)

为提升性能，OceanBase 数据库支持了多级的缓存系统，对于查询提供针对数据微块的 Block Cache，针对每个 SSTable 的 Row Cache，针对查询融合结果的 Fuse Row Cache，针对插入判空检查的 Bloomfilter cache 等，同一个租户下的所有缓存共享内存，当 MemTable 写入速度过快时，可以灵活的从当前各种缓存对象中挤占内存给写入使用。

## 多租户 (Multi-tenant)

在 OceanBase 数据库中，每一个租户即一个实例（类比 MySQL Instance），在一个 OceanBase 数据库中允许创建多个实例，即多租户。

## 1.5 E

## 二级索引 (Secondary Index)

访问数据表的一种辅助数据结构。与主键不同，二级索引通常包括由用户显式或隐式指定的一组键值。在 OceanBase 数据库中，二级索引一般实现为与主表关联的数据表。

## 1.6 F

### 反向增量（Reverse Incremental）

反向增量是指在数据迁移过程中，从目标系统到源系统传输数据的一种方式，即将在目标系统上发生的更改反向传输回源系统。

### 访问路径（Access Path）

数据库中访问某张表的特定方式，通常分为主键访问和二级索引访问。

### 分布式事务（Distributed Transaction）

OceanBase 数据库的事务类型由事务 Session 位置和事务涉及的日志流 Leader 数量两个维度来决定，主要分为分布式事务和单日志流事务。

以下两种场景的事务均称为分布式事务：

- 事务涉及的日志流数量大于一个。
- 事务涉及的日志流数量只有一个，且日志流的 Leader 和事务 Session 位置不在同一个 server 上。

### 分布式计划（Distributed Plan）

当执行计划涉及到多表或多分区时，该计划为分布式计划。

### 分布式执行（Distributed Execution）

执行计划在多台数据库服务器上执行，每台服务器完成其中的一部分工作。

### 分片（Tablet）

OceanBase 数据库 V4.0.0 版本引入了 Tablet 概念来表示实际的数据存储对象。它具备存储数据的能力，支持在机器之间迁移（transfer），是数据均衡的最小单位。Tablet 与分区一一对应，单分区表会创建一个 Tablet，多分区表会为每个分区创建一个 Tablet。索引表的每个分区也会对应一个 Tablet，包括局部索引表和全局索引表。特别地，局部索引表的 Tablet 与主表 Tablet 会强制绑定，保证存储在一台机器上。

### 分区（Partition）

与 Oracle 中 Partition 的概念相同，在 OceanBase 数据库中只有水平分区，表的每一个分区包含一部分记录。根据行数据到分区的映射关系不同，分为 Hash 分区、Range 分区（按

范围) 和 List 分区等。每一个分区, 还可以用不同的维度再分为若干分区, 叫做二级分区。例如, 交易记录表, 按照用户 ID 分为若干 Hash 分区, 每个一级 Hash 分区再按照交易时间分为若干二级 Range 分区。

## 分区表 (Partition Table)

OBServer 可以将普通表的数据按照一定规则划分到不同区块内, 同一区块内的数据物理上存储在一起, 这种划分区块的表叫做"分区表"。

## 分区裁剪 (Partition Pruning)

数据库根据查询条件, 避免访问无关分区的优化过程。分区裁剪可以分为静态和动态两种类型。

## 分区键 (Partition Key)

数据表的每一行中用于计算这一行属于哪个分区的列的集合, 叫做分区键。当前分区表支持一级分区和二级分区两种。由分区键构成的用于计算该行属于哪个分区的表达式, 叫做分区表达式。

如果是非分区表, 那么一张表对应一个分区; 如果是分区表, 则一张表对应多个分区。

分区类型支持: KEY、HASH、LIST、RANGE。

分区键数据类型支持: 数值、字符串、Date、Timestamp、二进制、ROWID。

## 分区副本 (Partition replica)

在数据库节点 (OBServer 节点) 组成的集群中, 所有的数据以分区为单位存储并提供高可用的服务能力, 每个分区有多个副本, 称为分区副本。

## 1.7 分区转移 (Transfer)

分区转移 (transfer) 是指将分区从一个日志流转移到另一个日志流。配合日志流的迁移等能力, 用以实现租户内资源扩缩容和负载均衡。

## 1.8 分页保序

分页保序是通过对查询结果进行排序, 然后再进行分页处理的方式来保持查询结果的顺序一致。对于给定的查询语句, 首先确定排序规则, 通常是基于一个或多个字段进行排序, 然后在原始查询语句的基础上添加 `ORDER BY` 子句, 按照排序规则对查询结果进行排序。最后再使用 `LIMIT` 子句来指定分页的起始位置和返回的记录数量。这样就能确保每次查询结果的顺序是一致的, 并按需返回指定页的记录。

## 副本 (Replica (V2.x/V3.x 版本))

为了数据安全和提供高可用的数据服务，每个分区数据在物理上存储多份，每一份叫做分区的一个副本。每个副本，包括存储在磁盘上的静态数据（SSTable）、存储在内存的增量数据（MemTable）、以及记录事务的日志三类主要的数据。根据存储数据种类的不同，副本有几种不同的类型，以支持不同业务在数据安全、性能伸缩性、可用性、成本等之间的选择。

- 全能型副本：也就是目前支持的普通副本，拥有事务日志，MemTable 和 SSTable 等全部完整的数据和功能。它可以随时快速切换为 Leader 对外提供服务。
- 日志型副本：只包含日志的副本，没有 MemTable 和 SSTable。它参与日志投票并对外提供日志服务，可以参与其他副本的恢复，但自己不能变为主提供数据库服务。
- 加密投票型副本：本质上是加密后的日志型副本，没有 MemTable 和 SSTable。它参与日志投票并对外提供日志服务，可以参与其他副本的恢复，但自己不能变为主提供数据库服务。
- 只读型副本：包含完整的日志，MemTable 和 SSTable 等，但是它的日志比较特殊。它不作为 Paxos 成员参与日志的投票，而是作为一个观察者实时追赶 Paxos 成员的日志，并在本地回放。这种副本可以在业务对读取数据的一致性要求不高的时候提供只读服务。因其不加入 Paxos 成员组，又不会造成投票成员增加导致事务提交延时的增加。

OceanBase 数据库 V2.x/V3.x 支持的副本类型如下表所示。

类型	LOG	Mem Table	SSTable	数据安全	恢复为 Leader 时间	资源成本	服务	副本类型转换限制	名称 (简写)
全能型	有，参与投票 (SYNC_CLOG)	有 (WITH_MEMSTORE)	有 (WITH_SSSTORE)	高	快	高	Leader 提供读写，Follower 可非一致性读	可转换为除加密投票型副本以外的任意副本类型	FULL(F)
日志型	有，参与投票 (SYNC_CLOG)	无 (WITHOUT_MEMSTORE)	无 (WITHOUT_SSSTORE)	低	不支持	低	不可读写	不能转换为其他副本类型	LOGONLY(L)
加密投票型	有，参与投票 (SYNC_CLOG)	无 (WITHOUT_MEMSTORE)	无 (WITHOUT_SSSTORE)	高	不支持	低	不可读写	不能转换为其他副本类型	Encrypt Vote (E)
只读型	有，异步日志，但不属于 Paxos 组，只是 listener (ASYNC_CLOG)	有 (WITH_MEMSTORE)	有 (WITH_SSSTORE)	中	不支持	高	可非一致性读	只能转换为全能型副本	READONLY(R)

副本根据负载和特定的策略，由系统自动调度分散在多个 Server 上。副本支持迁移、复制、增删、类型转换等管理操作。

## 副本 (Replica (V4.x 版本))

为了数据安全和提供高可用的数据服务，每个日志流的数据在物理上存储多份，每一份叫做日志流的一个副本。每个副本，包括存储在磁盘上的 Tablet 的静态数据 (SSTable)、存储在

内存上的 Tablet 的增量数据（MemTable）、以及记录事务的日志三类主要的数据。根据存储数据种类的不同，副本有几种不同的类型，以支持不同业务在数据安全、性能伸缩性、可用性、成本等之间的选择。

## 注意

OceanBase 数据库 V4.0.0 和 V4.1.0 版本仅支持全能型副本；V4.2.0 版本开始支持只读型副本；V4.3.3 版本开始支持列存副本。

- 全能型副本：也就是目前支持的普通副本，拥有事务日志，MemTable 和 SSTable 等全部完整的数据和功能。它可以随时快速切换为 Leader 对外提供服务。
- 只读型副本：包含完整的日志，MemTable 和 SSTable 等，但是它的日志比较特殊。它不作为 Paxos 成员参与日志的投票，而是作为一个观察者实时追赶 Paxos 成员的日志，并在本地回放。这种副本可以在业务对读取数据的一致性要求不高的时候提供只读服务。因其不加入 Paxos 成员组，又不会造成投票成员增加导致事务提交延时的增加。
- 列存副本：与只读副本类似，含完整的日志，MemTable 和 SSTable 等。不作为 Paxos 成员参与日志的投票，不可构成 Paxos 成员组。

相较于只读副本，列存副本最显著的特点就是用户表的基线数据使用列存方式存储，这里的用户表包括复制表，但不包括索引表、内部表以及系统表等。



OceanBase 数据库 V4.x 支持的副本类型如下表所示。

类型	LOG	Mem Table	SSTable	数据安全	恢复为 Leader 时间	资源成本	服务	副本类型转换限制	名称 (简写)
全能型	有, 参与投票 (SYNC_CLOG)	有 (WITH_MEMSTORE)	有 (WITH_SSSTORE)	高	快	高	Leader 提供读写, Follower 可非一致性读	可转换为只读型副本	FULL(F)
只读型	有, 异步日志, 但不属于 Paxos 组, 只是 listener (ASYNC_CLOG)	有 (WITH_MEMSTORE)	有 (WITH_SSSTORE)	中	不支持	高	可非一致性读	只能转换为全能型副本	READONLY(R)
列存	有, 异步日志, 但不属于 Paxos 组, 只是 listener (ASYNC_CLOG)	有 (WITH_MEMSTORE)	有 (WITH_SSSTORE)	中	不支持	高	可非一致性读	不能转换为其他副本类型	COLUMNSTORE (C)

副本根据负载和特定的策略, 由系统自动调度分散在多个 Server 上。副本支持迁移、复制、增删、类型转换等管理操作。

## 负载均衡 (Load Balance)

系统根据一定的策略, 通过动态调整 Unit 的位置和 Unit 内副本的位置, 使得同一个 Zone 内所有 server 的资源使用率达到均衡的过程。负载均衡策略要考虑很多因素, 目前分为两级调度。

具体信息请参见本节 **Partition 调度 (Partition Scheduling)** 和 **Unit 调度 (Unit Scheduling)** 的信息。



## 复制（Replication）

一种数据库复制技术，可以将数据库中的写入操作复制到多个从节点上，从而提高数据库的读取性能和可用性。

## 复制表（Replicated Table）

为应对应用访问频率高更新低频率同时总能访问到最新数据的小表访问需求，同时要保证数据一致性，目前只能选择强一致性读访问 Leader 数据的方案。但由于访问频率高，Leader 容易成为性能瓶颈。为了解决"小表广播"需求场景问题，OceanBase 数据库结合自身架构提供了复制表功能，将相关小表的副本复制到表所属租户的所有 OBServer 上。该表称为复制表，这些副本称为复制副本。复制表的更新事务在提交时保证将数据同步到所有的全功能副本及复制副本，确保在更新事务 commit 成功之后，在租户任意 OBServer 上能读到该事务修改的数据。

## Fuse Row Cache

在 LSM-Tree 架构中，同一行的修改可能存在于不同的 SSTable 中，OceanBase 数据库为了进一步优化存储占用，每次用户的更新都只会存储增量数据，因此在查询时需要对各 SSTable 查询的结果进行融合，当用户不再触发新的更新时，这个融合结果对查询都是一直有效的，因此 OceanBase 数据库也提供了对于融合结果缓存的 Fuse Row Cache，更大幅度支持部分用户的热点行查询。该功能类似于 Oracle 的 Result Cache 和 MySQL 的 Query Cache，都可以在查询相同数据时快速返回结果，从而减少数据库的负载和提升查询性能。

## 1.9 G

## 高可用（High Availability）

一个系统或服务在一定时间内保持高可用性的能力，即系统或服务能够持续地、无中断地运行并提供所需的服务。

## Granule

分布式并行执行计划中，对于表和索引扫描的最小工作任务粒度，可以是一个分区，或者是一个 query range。

## 工作线程（Worker Threads）

OceanBase 数据库中用以处理租户请求的线程。属于同一个租户的一组工作线程通过共享一个任务队列来服务用户的请求。

## 归一化（Normalization）

在机器学习领域中，数值数据转换或缩放为一个通用的范围，通常是 0 到 1 之间。归一化的目的是改变数值数据的表示方式。在 OceanBase 数据库中，归一化主要表示参数化后，将多个条件参数列表集合为一个单一的参数列表，用于 IN 子句查询，即将多个 IN 参数数组缩放为一个 IN 参数数组。

## 故障切换（Failover）

数据库主备库进行切换的一种切换方式。当数据库提供服务的主库出现故障或意外导致无法提供服务时，切换提供服务的集群为物理备库，该过程不可逆。

## 1.10 H

## Hint

数据库中用以直接指定优化器行为的用户原语。

## 行（Row）

一个行由若干列组成，有些时候其中的部分列构成主键（Row Key）并且整个表按主键顺序存储。有些表（例如，SELECT 指令的结果）可以不包含主键。

## 行版本合并（Row Compact）

将增量行数据中多个修改版本的数据合并成一行数据。

## 行缓存（Row Cache）

针对每个 SSTable 缓存具体的数据行，在进行 Get/MultiGet 查询时，可以将对应查到的数据行放入 Row Cache，这样在下次走到对应行查询时就可以避免多次二分定位对行的查找。

## 合并/每日合并（Major Compaction）

合并也就是 Major Compaction，在 OceanBase 数据库中也叫每日合并，概念和其他 LSM-Tree 数据库稍有不同。顾名思义，这个概念诞生之初是希望这个动作放到每天凌晨 2 点左右整个集群做一次整体的 Compaction 动作。合并一般是由每个租户的 RS 根据写入状态或者用户设置发起调度，每个租户的每次合并都会选取一个全局的快照点，租户内所有的分区都会用这个快照点的数据做一次 Major Compaction，这样每次合并租户所有的数据都基于这个统一的快照点生成相应的 SSTable，通过这个机制不仅能帮助用户定期整合增量数据，提升读取性能，同时还提供了一个天然的数据校验点，通过全局的一致位点，OceanBase 数据库能够在内部对多副本以及主表索引表进行多维度的物理数据校验。

## 宏块（Macro Block）

OceanBase 数据库将磁盘切分为大小为 2MB 的定长数据块，称之为宏块（Macro Block），宏块是数据文件写 IO 的基本单位，每个 SSTable 就由若干个宏块构成，宏块 2MB 固定大小的长度不可更改，后续转储合并重用宏块以及复制迁移等任务都会以宏块为最基本粒度。

## 宏块分裂（Macro Block Split）

由于宏块范围内插入和更新数据导致空间不足，需要将数据存放到多个宏块中。

## 宏块合并（Macro Block Merge）

由于删除数据，相邻的几个宏块中的所有行可以在一个宏块中存放，把相邻的多个宏块转换成一个宏块的过程。

## 宏块空间回收（Macro Block Recycle）

合并过程中会根据原基线数据和变更数据生成新的基线数据，原基线数据的宏块被重复利用的过程称为回收。

## 宏块预读（Macro Block Prefetch）

在范围查询的过程中，根据需要提前预读取相邻的宏块。

## 活跃 MemTable（Active MemTable）

当前活跃的 MemTable，可以写入增量数据。与 Frozen MemTable 相对应。

## 活跃事务（Active Transaction）

指事务已经开启，但还没有提交或者回滚的事务。活跃事务所做的修改在提交前都是临时的，别的事务无法看到。

## 1.11 J

## 集群（Cluster）

OceanBase 数据库集群由一个或多个 Region 组成，Region 由一个或多个 Zone 组成，Zone 由一个或多个节点组成，每个节点可有若干个 Unit，每个 Unit 可有若干个日志流 Logstream 的 Replica，每个 Logstream 可使用若干个 Tablet。

## 集群实例（Cluster Edition）

提供集群模式的 OceanBase 云数据库服务，规格支持 4C/8C/14C/24C/30C/62C，默认为 3 副本方式（包括 3F 和 2F1L），支持 MySQL 和 Oracle 模式。

## 基表（Base Table）

基表是物化视图所依赖的源表，物化视图的数据来自于这些表的查询结果。

## 基线数据（Baseline Data）

每日合并生成的存储于持久化介质上的只读有序数据。

## 基线数据版本（Baseline Data Version）

基线数据的版本。

## 级联复制（Cascading）

指一种数据复制和同步技术，它可以将一个数据库的更新操作自动复制到多个从节点，从而实现数据的分发和同步。

## 渐进合并（Progressive Compaction）

为了降低全量合并对系统的影响，一轮合并操作只合并部分的宏块，经过几轮以后，完成所有宏块的合并。采用渐进合并的主要目的是控制单次合并的时间，在表模式发生变化（如增加列、修改压缩算法）的时候，需要在合并过程中更新所有行，如果是一个大表，更新所有行对合并时间有很大的影响。在这种情况下，采用渐进合并能有效控制每次合并的时间。

## 监控系统（Monitoring System）

常见的监控系统会涉及到资源水位的监控和应用状态的监控，资源监控主要面向基础云资源例如 ECS/RDS 的 CPU/MEM/IO 等等的监控，应用监控主要面向的各应用服务的健康状态，以及各应用系统特有的一些指标，例如 RPC 调用/error 计数/PV 活跃度等等维度的统计。

## OBServer 节点，节点（OBServer Node, Node）

OceanBase 数据库服务器。server 是运行 observer 进程的物理机，一台物理机上可以部署一个或者多个 OBServer。在 OceanBase 数据库内部，server 由其 IP 地址和服务端口唯一标识。

## 1.12 K

## 可用区/区（Availability Zone, AZ）

AZ 指的是"可用区"（Availability Zone），是指在同一地理区域内，相互隔离的数据中心集群。可用区通常由相互独立的电力、网络 and 冷却系统组成，以确保高可用性和冗余性。

AZ 通常用于云计算平台（如 AWS、Azure 等），以提供高可用性和容错性。在同一地理区域内选择不同的可用区部署应用程序和数据可以确保在某些不可避免的故障（如硬件故障、自然灾害等）发生时，服务可以继续运行。

具体到 AWS 上的可用区，它们是由 AWS 在同一地理区域内，分别建立在不同的数据中心，以确保各个可用区之间的独立性，从而提高可用性和容错性。在 AWS 中，每个可用区都具有自己的电力、网络 and 冷却系统，并且与其他可用区相互隔离。在同一地理区域内使用不同的可用区可以确保高可用性，减少故障对应用程序的影响。

## 库，数据库（Database）

数据库对象，一个数据库对象中可以包含表、视图等其他数据库对象。

## 快速参数化（Fast Parsing）

OceanBase 数据库特有的针对实参 SQL 快速扣取参数（参数化）的过程。快速参数化结合 OceanBase 数据库计划缓存的固有特点，通过增加约束条件等方法，避免了输入 SQL 再次执行时的语义分析过程，加速了计划匹配的效率。

### 1.13 L

## Label Security

是强制访问控制的一种方式，通过在表中添加一个 Label 列来记录每行的 Label 值，在访问时通过比较用户的 Label 和数据的 Label，达到约束主体（用户）对客体（表中的数据）访问的目的。

## Learner

指的是不参与日志多数派同步的副本，例如，只读副本。

## Liboblog

Liboblog 是 OceanBase 数据库的增量数据同步工具，通过 RPC 方式拉取 OceanBase 数据库各个分区的 Redo 日志后，结合各个表和列的 Schema 信息，转换 Redo 日志为中间定义的数据格式，最后以事务的方式输出修改的数据。

## 联接顺序（Join Order）

多表联接时各表之间的联接顺序。

## 联接算法（Join Algorithm）

执行两表联接时使用的算法，包括 Nested Loop Join、Merge Join 和 Hash Join 三种。

## 链路（Pipeline）

数据处理管道，是一种将多个数据处理操作连接在一起的方式，从而实现数据的转换、处理和传输的过程。

## Location Cache

OceanBase 数据库按日志流、分片组织用户数据，且每个日志流有多个副本用于容灾。因此，在 SQL 请求执行过程中需要知道分区数据的位置信息，用于路由到特定机器读写对应副本的数据。分区数据的位置信息即称为 Location，每个 observer 进程会有一个服务，用于刷新及缓存本机需要的分区 Location 信息，该服务称为 Location Cache 服务。

## Log Group

在 OceanBase 数据库中，每一条归档日志实际上是一个日志集合，包含若干条 Log Entry，该条归档日志称之为 Log Group。每个 Log Entry 都有一个 SCN 与之关联，Log Group 也有一个 SCN，是所有 Log Entry 中最大的 SCN。日志归档的工作就是在归档介质上管理组织 Log Group。

## 轮转合并（Rotating Major Compaction）

为了减少合并对业务的影响，各个 Zone 按照指定的顺序依次合并，正在合并的 Zone 不对外提供服务。

## 逻辑数据中心（Logical Data Center, LDC）

LDC 是对 IDC（Internet Data Center，互联网数据中心）的一种逻辑划分。OceanBase 数据库在多地多中心部署时，会以 region 和 Zone 的单位对所有 OBSERVER 服务器进行划分，此时 OceanBase 数据库的客户端路由和 OceanBase 数据库内部的 RPC 路由被称为 LDC 路由。

## 1.14 M

## MaaS（Management as a service）

管理运维即服务，在客户自有 IAAS 等基础设施上采购和部署数据库软件，并向数据库云厂商采购运维管理服务。

## Meta 租户（Meta Tenant）

Meta 租户是 OceanBase 数据库内部自管理的租户，每创建一个用户租户会创建一个对应的 Meta 租户，其生命周期与用户租户保持一致。Meta 租户用于存储和管理用户租户的集群私有数据，这部分数据不需要跨库物理同步以及物理备份恢复，例如：配置项、位置信息、副本信息、日志流状态、备份恢复相关信息、合并信息等。Meta 租户不可登录，普通用户只能通



过系统租户的视图查询 Meta 租户下的数据。Meta 租户没有独立的 Unit，创建租户时默认为 Meta 租户预留资源，各项资源从用户租户资源中扣除。

## MVCC (Multi-version Concurrency Control)

多版本并发控制。

## 慢查询 (Slow Query)

超过指定时间的 SQL 语句查询。

## Membership Log

一种特殊的记录一个分区成员组变更记录的 commit log。

## Mini SSTable

L0 层 SSTable。根据不同转储策略需要的不同参数设置，L0 层 SSTable 可能存在可以为空。对于 L0 层提供 server 级配置参数来设置 L0 层内部分层数和每层最大 SSTable 个数，L0 层内部即分为 level-0 到 level-n 层，每层最大容纳 SSTable 个数相同。当 L0 层 level-n 的 SSTable 到达一定数目上限或阈值后开始整体 compaction，合并成一个 SSTable 写入 level-n+1 层。当 L0 层 max level 内 SSTable 个数达到上限后，开始做 L0 层到 L1 层的整体 compaction 释放空间。在存在 L0 层的转储策略下，冻结 MemTable 直接转储在 L0-level0 写入一个新的 Mini SSTable，L0 层每个 level 内多个 SSTable 根据 base\_version 有序，后续本层或跨层合并时需要保持一个原则，参与合并的所有 SSTable 的 version 必须邻接，这样新合并后的 SSTable 之间仍然能维持 version 有序，简化后续读取合并逻辑。

L0 层内部分层会延缓到 L1 的 compaction，更好的降低写放大，但同时会带来读放大，假设共 n 层，每层最多 m 个 SSTable，则最差情况 L0 层会需要持有  $(n \times m + 2)$  个 SSTable，因此实际应用中层数和每层 SSTable 上限都需要控制在合理范围。

## Minor SSTable

L1 层内部称为 Minor SSTable，L1 层的 Minor SSTable 仍然维持 rowkey 有序，每当 L0 层 Mini SSTable 达到 compaction 阈值后，L1 层 Minor SSTable 开始参与和 L0 层的 compaction。为了尽可能提升 L1 Compaction 效率，降低整体写放大，OceanBase 数据库内部提供写放大系数设置，当 L0 层 Mini SSTable 总大小和 L1 Minor SSTable 大小比率达到指定阈值后，才开始调度 L1 Compaction，否则仍调度 L0 层内部 Compaction。

## Major SSTable

L2 层是基线 Major SSTable，为保持多副本间基线数据完全一致，日常转储过程中 Major SSTable 仍保持只读，不发生实际 compaction 动作。

## Multi-Paxos

一种执行多 Paxos 实例的优化协议，OceanBase 数据库使用 multi-Paxos 协议实现 commit log 的多机持久化。

## 模糊绑定 OUTLINE

模糊绑定使数据库能够识别和应用一个大致匹配的执行计划（即 Outline）到一组模糊相似的查询上，而不是精确匹配。

OceanBase 数据库的模糊绑定 OUTLINE 指的是多个 SQL 共享一个 OUTLINE。

## MySQL 模式（MySQL Mode）

MySQL 兼容性模式。为降低 MySQL 数据库迁移 OceanBase 数据库引发的业务系统改造成本，使业务数据库设计人员、开发人员、数据库管理员等可复用积累的技术知识经验，快速上手使用 OceanBase 数据库所做的一种租户类型功能，目前高度兼容 MySQL 语法，常用系统表、函数保持一致。

## mysqltest

数据库的测试小工具，用于测试数据库是否按照预期运行，包括一组测试用例和相关运行程序。

## 1.15 N

## Nop Log

commit log 中的一种特定类型的日志，表示空操作。nop log 产生于 multi-Paxos 协议的恢复阶段，如果一条日志没有多数派副本上持久化成功，在恢复阶段就可能产生一条 nop log 作为这条日志的内容。

## 1.16 O

## OBKV-Table

OBKV-Table 是 OceanBase 提供的一组对表模型数据进行读写的接口，在 OceanBase 内部定义了一组用于 OBKV-Table 的客户端和数据库服务端之间通用的交互协议，用户通过 OBKV-Table 提供的 SDK 可以直接对 OceanBase 数据库中的关系表数据读写，OBKV-Table 直接访问 OceanBase 存储层和事务层，能够为用户提供更加高效的数据读写能力。

## OBKV-HBase



OceanBase 的 OBKV-HBase 客户端基于 OBKV-Table 提供的基本接口，在客户端封装了 HBase 兼容的 API 接口（目前已兼容 HBase 0.94 版本的特性）。若您当前有业务使用了原生 HBase 数据操作逻辑，您可以通过安装 OceanBase 数据库集群，在 OBServer 服务端建立 HBase Table，并通过 OBKV-HBase 进行数据操作。

## OBAgent

OBAgent 是一个监控采集框架。OBAgent 支持推、拉两种数据采集模式，可以满足不同的应用场景。OBAgent 默认支持的插件包括主机数据采集、OceanBase 数据库指标的采集、监控数据标签处理和 Prometheus 协议的 HTTP 服务。要使 OBAgent 支持其他数据源的采集，或者自定义数据的处理流程，您只需要开发对应的插件即可。

## ob\_admin

OceanBase 数据库的配套运维工具。ob\_admin 提供了 slog\_tool、clog\_tool、dumpsst 和 dump\_backup 功能，主要用于排查数据不一致、丢数据、错误数据等问题。

## ob\_error

OceanBase 数据库的一个错误码解析工具，ob\_error 可以根据您输入的错误码返回相对应的原因和解决方案。

## observer 进程（Observer Process）

OceanBase 数据库是单进程软件，进程名为 observer。通常一台物理或者虚拟服务器运行一个 observer 进程，由 IP 和端口作为唯一标识，我们称之为节点，也可以用 OBServer 节点表示。observer 进程作为 OceanBase 数据库最核心的组件，负责几乎所有数据库内核功能，包括 SQL 引擎、存储引擎和事务引擎。分布式的功能也同样在这个进程中，包括 RPC 通信、分区管理和负载均衡等。

## OBServer 服务器（OBServer Host）

用来安装部署 OBServer 节点的物理机。

## OceanBase 安装部署工具（OceanBase Deployer）

是 OceanBase 集群安装部署工具，通过命令行部署或白屏界面部署的方式，将复杂配置流程标准化，降低集群部署难度。

## OceanBase C API 库（OceanBase Connector /C）

OceanBase Connector/C 是一个基于 C/C++ 的 OceanBase 客户端开发组件，支持 C API Lib 库。允许 C/C++ 程序以一种较为底层的方式访问 OceanBase 分布式数据库集群，以进行数据库连接、数据访问、错误处理和 Prepared Statement 处理等操作。

## OceanBase Cloud

构建在阿里云、AWS 等全球主流公有云基础设施上，基于完全自主研发的原生分布式数据库，提供弹性扩展、卓越性能、主流兼容的高性价比的数据库云服务。为客户在云上提供监控、诊断、开发、迁移、备份、恢复的端到端数据库服务化解决方案。

## OceanBase 导数工具 (OceanBase Loader and Dumper)

一款使用 Java 开发的客户端导入工具。该工具提供了非常灵活的命令行选项，可在多种复杂的场景下，将定义和数据导入到 OceanBase 中。推荐 obloader 与 obdumper 搭配使用，但是在外部业务中，obloader 同时支持如 Navicat、Mydumper 和 SQLDeveloper 等工具导出的 CSV 格式的文件导入。obloader 充分利用 OceanBase 分布式系统的特性，重点优化导入性能和稳定性，以及丰富运行监控信息提升用户体验。

## OceanBase 管理者工具 (OceanBase Admin Toolkit, OAT)

一个可视化平台，用于安装和管理 OceanBase 生态产品和组件。OceanBase 生态产品包括：OceanBase 云平台 (OceanBase Cloud Platform, OCP)、OceanBase 开发者中心 (OceanBase Development Center, ODC)、OceanBase 访问代理 (OB Sharding) 以及 OceanBase 迁移服务 (OceanBase Migration Service, OMS)。OceanBase 组件包括：MetaDB、OBDNS、InfluxDB 和 NLB。

## OceanBase JDBC 驱动程序 (OceanBase Connector/J)

JDBC (Java Database Connectivity) 是 Java 应用程序访问数据库的标准 API (应用程序编程接口)，数据库驱动的实现会将 JDBC 标准编程接口转换成对应数据库厂商的 SQL 实现。OceanBase 数据库 JDBC 驱动兼容 JDBC 4.0、4.1、4.2 标准。OceanBase 提供了自研的驱动，使用该驱动可以同时使用 OceanBase 数据库的 MySQL 模式和 Oracle 模式。

## OceanBase 开发者中心 (OceanBase Developer Center, ODC)

为 OceanBase 数据库量身打造的企业级数据库开发平台。ODC 支持连接 OceanBase 中 MySQL 和 Oracle 模式下的数据库，同时为数据库开发者提供了数据库日常开发操作、WebSQL、SQL 诊断、会话管理和数据导入导出等功能。

## **OceanBase 命令行客户端（OceanBase Command-Line Client, OBClient）**

OBClient 是 OceanBase 数据库命令行客户端，可以访问 OceanBase 数据库的 MySQL 租户和 Oracle 租户。

## **OceanBase OCI 驱动程序（OceanBase Call Interface, OBCI）**

OBCI 驱动是兼容 Oracle 数据库 OCI 接口的 C 语言的驱动。基于 Oracle OCI 开发的用户和应用可以使用 OBCI 驱动，平滑的迁移使用 OceanBase 数据库的 Oracle 模式。

## **OceanBase ODBC 驱动程序（OceanBase Connector/ODBC）**

开放数据库连接（Open Database Connectivity, ODBC）是为解决异构数据库间的数据共享而产生的，现已成为 WOSA（The Windows Open System Architecture），即 Windows 开放系统体系结构的主要部分和基于 Windows 环境的一种数据库访问接口标准。ODBC 为异构数据库访问提供统一接口，允许应用程序以 SQL 为数据存取标准，存取不同 DBMS 管理的数据；使应用程序直接操纵数据库中的数据，免除随数据库的改变而改变。用 ODBC 可以访问各类计算机上的数据库文件，也可以访问例如 Excel 表和 ASCII 数据文件等非数据库对象。

## **OceanBase 嵌入式 SQL 预编译器（Embedded SQL in C for OceanBase, ECOB）**

ECOB 是与 Oracle ProC 兼容的 OceanBase 预编译器，提供了与 ProC 兼容的功能特性。ECOB 由预编译器程序 ecob 和运行时动态链接库 ecoblib（libecob.so）组成。

## **OceanBase 迁移服务（OceanBase Migration Service, OMS）**

OceanBase 数据库提供了一种支持同构或异构数据源与 OceanBase 数据库之间进行数据交互的服务，具备在线迁移存量数据和实时同步增量数据的能力。

## OceanBase 迁移评估 (OceanBase Migration Assessment, OMA)

OceanBase 提供的数据库迁移评估的产品，为数据迁移提供精准的兼容性评估、高效的性能评估以及应用逻辑改造建议。OMA 支持评估 Oracle、DB2 LUW、PostgreSQL 等多种数据库与 OceanBase 的兼容情况，提供画像分析和自动转换方案；支持应用负载回放功能，帮助客户预知迁移后可能的性能风险并提供优化方案；OMA 还支持评估 C、Java 业务代码以及驱动的兼容性以助力用户高效率、低成本迁移至 OceanBase。

## OceanBase 数据库 (OceanBase Database)

一款完全自研的企业级原生分布式数据库，在普通硬件上实现金融级高可用，首创“三地五中心”城市级故障自动无损容灾新标准，刷新 TPC-C 标准测试，单集群规模超过 1500 节点，具有云原生、强一致性、高度兼容 Oracle/MySQL 等特性。

## OceanBase 数据库企业版 (OceanBase Database Enterprise Edition)

OceanBase 企业版是一款完全自研的企业级原生分布式数据库，在普通硬件上实现金融级高可用，首创“三地五中心”城市级故障自动无损容灾新标准，刷新 TPC-C 标准测试，单集群规模超过 1500 节点，具有云原生、强一致性、高度兼容 Oracle/MySQL 等特性。

## OceanBase 数据库社区版 (OceanBase Database Community Edition)

OceanBase 数据库社区版是兼容 MySQL 的单机分布式一体化国产开源数据库，具有原生分布式架构，支持金融级高可用、透明水平扩展、分布式事务、多租户和语法兼容等企业级特性。OceanBase 内核通过大规模商用场景的考验，已服务众多行业客户；面向未来携手社区生态伙伴，共建开源开放的数据库内核和生态。

## OceanBase 数据库代理 (OceanBase Database Proxy, ODP)

OceanBase 数据库专用的代理服务器，OceanBase 数据库用户的数据会以多副本的形式存放在各个 OBServer 节点上，ODP 接收用户发出的 SQL 请求，并将 SQL 请求转发至最佳目标 OBServer 节点，最后将执行结果返回给用户。

## OceanBase 云平台 (OceanBase Cloud Platform, OCP)

一款以 OceanBase 为核心的企业级数据库管理平台。OCP 不仅提供对 OceanBase 集群和租户等组件的全生命周期管理服务，同时也对 OceanBase 相关的资源（主机、网络和软件包等）提供管理服务，让您能够更加高效地管理 OceanBase 集群，降低企业的 IT 运维成本。

## OceanBase 云平台 Express (OCP Express)

OCP Express 是一个基于 Web 的 OceanBase 4.x 管理工具，它集成在 OceanBase 数据库集群中，支持数据库集群关键性能指标查看及基本数据库管理功能。OCP Express 脱胎于 OceanBase 云平台（OceanBase Cloud Platform, OCP），在保留云平台核心能力的基础上，调整了整体的功能布局，给予用户全新的使用体验；同时也进一步调整了功能配置，使得 OCP Express 可以以极小的资源消耗部署在任意一台数据库节点上，使得 OCP Express 用户以最小的成本，获取全新的 OceanBase 4.x 管控体验。

## OLAP（On-line Analytical Processing）

联机分析处理。

## OLTP（On-line Transaction Processing）

联机事务处理。

## Oracle 模式（Oracle Mode）

Oracle 兼容性模式。为降低 Oracle 数据库迁移 OceanBase 数据库引发的业务系统改造成本，使业务数据库设计人员、开发人员、数据库管理员等可复用积累的技术知识经验，快速上手使用 OceanBase 数据库所做的一种租户类型功能，逐步兼容 Oracle 语法，常用系统表、函数保持一致。

## 1.17 P

## Partition 调度（Partition Scheduling）

对于每一个租户每个 Zone 中的若干个 UNIT，通过在 Unit 之间迁移副本，达到每个 Unit 内资源使用率的均衡。

其中：

- 属于同一个分区表的若干不同分区，会均匀分散在不同的 Unit 上，使得每个 Unit 有相同个数的分区。
- 属于同一个 partition group 的若干分区，会聚集在同一个 Unit 上。
- 在保证上述每组分区个数均衡的基础上，通过交换这组分区内的分区，降低 Server 的磁盘水位线。
- 通过迁移非分区表的分区，降低 Server 的磁盘水位线。



## Piece

OceanBase 数据库按照 Piece 来组织和管理归档日志数据，一个 Piece 是某个租户连续一段时间内完备日志的集合，是一个左闭右开的区间。

## PX worker

分布式并行执行下，在每一个机器上参与执行一个分布式并行计划的工作线程。

## 1.18 Q

## Query Coordinator (QC)

在分布式并行执行计划中，主控节点上的一个线程，用于调度、协调整体分布式并行执行计划的执行。

## 迁入 (Move In)

Move In 是用户通过 Schema 变更把一个没有表组 (Table Group) 属性的表格设置上表组属性。数据库内会检查被操作表格的分区方式和目标表组的分区方式，只有完全一致的情况，DDL 才会成功。

## 迁出 (Move Out)

迁出是迁入的逆操作，用户通过 Schema 变更把一个表格的表组 (Table Group) 属性删除，即把这张表从表组内迁出。用户通过 Schema 变更把一个表格的表组属性从 A 修改为 B，表格会从 A 中迁出，然后迁入 B。

## 迁移 (Migration)

将分区的副本从一个节点迁到另一个节点，先在目标节点上增加一个副本，完成后再将源节点上的副本删除。

## 全链路追踪 (End to End Tracing)

全链路有两条路径，一条是从应用通过客户端 (JDBC 或 OCI 等) 下发请求给 ODP (代理服务器) 访问 OBServer 节点，访问结果返回给应用；另一条是从应用通过客户端 (JDBC 或 OCI 等) 直接访问 OBServer 节点，访问结果返回给应用。全链路追踪是对全链路所有组件进行问题定位的诊断。

## 全量合并 (Full Compaction)

进行全量合并时，无论分区的宏块是否修改都重新生成一遍。可以在表粒度指定是否采用全量合并，通常在表模式发生变化 (如增加、删除列) 或者存储属性发生变更 (如修改压缩级别)

时，对该表进行全量合并。在需要对表进行全量合并时，通常采用渐进合并的方式来控制单次合并的时间。

在全量合并过程中，会把当前的静态数据都读取出来，和内存中的动态数据合并后，再写到磁盘上去作为新的静态数据。在这个过程中，会把所有数据都重写一遍。全量合并会极大的耗费磁盘 IO 和空间，除了 DBA 强制指定外，目前 OceanBase 数据库一般不会主动做全量合并。

## 全量校验（Full Verification）

在数据备份和恢复过程中，将备份数据与原始数据进行完全的比较和校验的过程。在进行全量校验时，备份数据和原始数据会被逐一比较，以确保备份数据的完整性和准确性。

## 全量刷新（Complete Refresh）

全量刷新是指完全重新计算物化视图中的数据。

## 全量数据导入（Initial Load）

在数据库系统中，将一个完整的数据集合从一个数据源复制到另一个目标位置的过程。

## 全量提取（Initial Load Extract）

指在数据库系统中，从源数据库中提取出一个完整的数据集合的过程。通常，全量提取是在数据备份、数据复制或数据迁移等过程中进行的。

## 全局时间戳服务（Global Timestamp Service，简称 GTS）

OceanBase 数据库内部每个租户启动一个全局时间戳服务，事务提交时通过本租户的时间戳服务获取事务版本号，保证全局的事务顺序。

## 全局索引（GlobalIndex）

OceanBase 数据库中跨分区的数据索引。全局索引通过全局化的存储数据键值与主键信息，可以快速定位到数据所在的分区。

## 全局一致性快照（Global Consistent Snapshot）

在没有实现全局一致性快照前，分布式数据库无法实现跨节点的一致性读和保证因果序。OceanBase 数据库 V1.4.x 版本中，应用系统设计和开发人员需要保证一条 SQL 语句中访问的多个表、多个分区都在同一个 OceanBase 数据库节点上，同时对于依赖操作顺序的业务系统，无法保证两个前后事务分别修改两个节点上两张表的数据反映顺序。OceanBase 数据库

V2.0 版本实现的全局一致性快照功能，从根本上解决了这些问题。相对于 Google Truetime 基于原子钟的硬件实现，OceanBase 数据库的全局时间戳（GTS）服务是纯软件实现的，不依赖特定的硬件设备，也不对客户方的部署环境提额外的要求，使得 OceanBase 数据库能够服务更广泛的专有云客户。OceanBase 数据库 V2.x 版本的全局时间戳打开后，跨节点读写、因果序的行为和单机数据库完全一致。

## 强一致性读/弱一致性读（Strong Read Consistency/Weak Read Consistency）

强一致性读是默认的一种 OceanBase 数据库的 SQL 执行方式，即 SQL 必须转发到涉及 Partition 的 Leader 所在的 OBServer 服务器上才能执行，用以保证获取到实时最新数据。弱一致性读是相对于强一致性读来说的。即 SQL 只需在涉及 Partition 的 OBServer 服务器上执行即可，不强制是 Leader。如需使用弱一致性 SELECT，主要通过两种方式。一种是带 read\_consistency(weak) Hint 的 SELECT 语句，另一种是在当前 Session 中修改 ob\_read\_consistency 的系统变量取值为 weak。

## 1.19 R

## Redo 日志索引（Redo Log Index）

llog 操作日志（commit log）在文件中不是根据日志 ID 有序存放的，为顺序读取操作日志，在 ilog 中存放了每条操作日志在日志文件中的位置，ilog 中的记录是按照日志 ID 有序的。

## Rebuild

当一个副本落后且无法拉取到缺失的日志时（源端日志已经回收），通过拷贝基线数据的方式追赶的过程称为 Rebuild。

## 日志服务（Log Service）

简称 LOG 是针对日志类数据的一站式服务。使用者无需开发就能快捷完成日志数据采集、消费、投递以及查询分析等功能，提升运维、运营效率，建立 DT 时代海量日志处理能力，常见的日志系统的解决方案有基于 ELK STACK 的自建系统，也有选择成熟的云产品，例如阿里云的 SLS。

## 日志归档（Log Archive）

日志归档是指日志数据的自动归档功能，OBServer 会定期将日志数据归档到指定的备份路径。这个动作是全自动的，不需要外部定期触发。

## 日志流（Log Stream, LS）



日志流是由 OceanBase 数据库自动创建和管理的实体，它代表了一批数据的集合，包括若干 Tablet 和有序的 Redo 日志流。它通过 Paxos 协议实现了多副本日志同步，保证副本间数据的一致性，实现了数据的高可用。日志流副本支持在不同机器之间迁移（Migration）和复制（Replication），以达到机器管理和系统容灾的目的。

从数据存储的角度来看，日志流可以抽象为 Tablet 容器，支持添加和管理 Tablet 数据，允许 Tablet 在不同日志流之间转移（Transfer），以达到数据均衡和水平扩展的目的。

从事务的角度来看，日志流是事务的提交单位。事务修改在单个日志流内完成时可以采用一阶段原子提交；事务修改跨多个日志流时，采用 OceanBase 数据库优化的两阶段提交协议完成原子提交，日志流是分布式事务参与者。

## 日志流组（Log Stream Group, LS Group）

同一日志流的多个副本使用 Paxos 一致性协议保证副本的强一致，每个日志流和它的副本构成一个独立的 Paxos 组，其中一个日志流为主副本（Leader），其它日志流为从副本（Follower）。主副本具备强一致性读和写能力，从副本具备弱一致性读能力。

## 容器服务（Container Service）

提供高性能可伸缩的容器应用管理服务，支持用容器进行应用生命周期管理，提供多种应用发布方式和持续交付能力并支持微服务架构，其关联了若干服务器节点、负载均衡、专有网络等云资源；提供安全、高性能、支持混合云的部署方案；同时整合负载均衡，提供容器的访问能力；通过高可用调度策略，轻松打通上下游交付流程。

## 容错（Fault Tolerance）

是指一个系统、设备或软件能够在出现故障或错误的情况下，仍然能够继续正常运行或者最小化对系统的影响。

## Root Service（RS）

OceanBase 数据库集群会有一个总控服务（Root Service），其运行在某个 OBServer 节点上。当 Root Service 所在机器故障时，其余节点会选举出来新的 Root Service。Root Service 主要提供资源管理、容灾、负载均衡、schema 管理等功能。

## Rowid

分区局部索引的基线数据行中保存的主表行数据的位置信息，用于快速定位相应的主表行。

## Row Merge

将基线行数据和增量行数据进行融合形成新版本行的过程。

## RPC（Remote Process Call）

远程方法调用。

## RS List

OceanBase 集群中启动了 Root Service 服务的机器的 IP 列表（一般是每个 Zone 一个）。

RS List 涉及集群的创建，与 `sys` 租户关闭密切。

RS List 有两种集合形态，一种是直接从 Config Server 获取的 RS List；另一种是用 Server List 更新后的 RS List。

## Reconfirm

Reconfirm 即再次确认，Multi-Paxos 协议中 Leader 上任需要执行的一个流程。分区 Leader 上任之后，要把之前这个分区多数派上持久化成功的日志再次确认一遍，并把这些确认过的日志同步到所有正常工作的副本，确认多数派都持久化成功所有日志后 Reconfirm 才算成功。

## 1.20 S

## SaaS (Software-as-a-service)

软件即服务，是基于互联网提供软件服务的软件应用模式，在这里特指在公有云上，三方独立软件为客户提供软件服务的应用模式

## Savepoint

OceanBase 数据库提供的可以由用户定义的一个事务内的执行标记。用户可以通过在事务内定义若干标记并在需要时将事务恢复到指定标记时的状态。

## Schema

通常来说，指的是数据库对象（如表、视图、索引等）的模式；在 OceanBase 数据库服务端，也指所有数据库对象模式的集合。

## Schema Version

在 OceanBase 数据库服务端，维护了一个全局的 Schema 版本号，数据库对象的每次变化都会引起全局 Schema 版本的升高。每一个数据库对象也有一个自己的版本号，该版本号是该对象最近一次修改（创建、变更等）对应的全局 Schema 版本号。

## 闪回查询 (Flashback)

OceanBase 数据库提供了记录级别的闪回查询 (Flashback Query) 功能，该功能允许用户获取某个历史版本的数据。

## 视图（View）

视图是一种虚拟表，与包含数据的实际表不同，视图只保存了一个查询的结果集的结构。

## 数据编码（Encoding）

在通用压缩的基础上，OceanBase 数据库自研了一套对数据库进行行列混存编码的压缩方法（Encoding）。和通用压缩不同，Encoding 的基础建立在压缩算法感知数据块内部数据的格式和语义的基础上。OceanBase 数据库是一个关系型数据库，数据是以表的形式来组织的，表中的每一列都有固定的类型，这就保证了同一列数据在逻辑上存在着一定的相似性；而且在一些场景下，业务的表中相邻的行之间数据也可能会更相似，所以如果将数据按列进行压缩并存储在一起就可以带来更好的压缩效果。因此，OceanBase 数据库引入了一种 Encoding 格式的微块，与所有数据逐行序列化到块中的 flat 格式的微块不同，Encoding 格式的微块是行列混存的，逻辑上仍然是一组行的数据存在微块里，但微块会按列对数据进行编码，编码后的定长数据存储存储在微块内部的列存区，部分变长数据还是按行存储在变长区。而且在 Encoding 微块中，数据是可以随机访问的，当需要读微块中的一行数据时，可以只对这一行数据进行解码，避免了部分解压算法读一部分数据要解压整个数据块的计算放大；在向量化执行的过程中也可以对指定的列进行解码，降低了投影的开销。

## 数据订阅（Data Subscription）

数据订阅是一种机制，允许用户通过订阅特定的数据库对象（例如表、视图或查询）来实时或定期获取更新的数据。

## 数据复制（Data Replication）

指将一个数据库中的数据复制到其他数据库系统的过程。在数据复制过程中，数据会从源数据库中抽取出来，经过一系列的处理和传输，最终被复制到目标数据库中，以保证数据在不同系统之间的同步性和一致性。

## 数据获取（Data Ingestion）

将不同来源的数据导入到一个特定的系统或应用程序中，以进行处理、分析、可视化或存储。

## 数据集成（Data Integration）

数据集成是指将来自不同数据源、不同格式和不同位置的数据合并到一个一致的视图中的过程。

## 数据均衡（Data Balancing）

OceanBase 数据库通过 Root Service 管理租户内各个资源单元间的负载均衡。不同类型的副本对资源的需求各不相同，Root Service 在执行均衡操作时需要考虑的因素包括每个资源单元的 CPU、磁盘使用量、内存使用量以及 IOPS 使用情况。经过负载均衡，最终会使得所有机器的各类型资源占用都处于一种比较均衡的状态，充分利用每台机器的所有资源。

## 数据迁移（Data Migration）

数据迁移是指将数据从一个数据存储位置或系统移动到另一个位置或系统的过程。

## 数据转换（Data Transformation）

将原始数据按照一定规则进行处理和转换，以生成新的数据或提取数据特征，从而更好地进行分析、建模和可视化。

## Slog/SSTable Log

节点为了维护本机基线数据一致性而使用的日志。

## Server List

OceanBase 集群所有机器的 IP 列表。

## sql\_diagnoser

一款敏捷版 SQL 诊断工具，可直接分析业务集群，找出常见的可疑 SQL 及隐藏的性能问题。

## SQL Plan Management (SPM)

一种计划演进的机制。当优化器生成新的计划时，需要通过演进机制来保证这个计划的性能不会出现回退，如果出现回退，就拒绝使用该计划，否则使用该计划。

## Sorted Strings Table (SSTable)

用于存储基线数据或转储数据，行数据有序存储。

## Sub Query Coordinator (SQC)

分布式并行执行计划中，每一台参与执行的服务器上会有一个本地的协调者，用于接收从 QC 发送来的调度指令，获取本地执行工作线程，生成本地执行任务的 granule，协调本地执行。

## 数据库（Database）

数据库是按数据结构来组织、存储和管理数据的仓库。数据库下包括若干表、索引，以及数据库对象的元数据信息。

## 数据倾斜（Data Skew）

数据中对于某一个或某几个值出现的次数特别多，占据了对应数据较大的比例，在分布式执行过程中，数据倾斜会引起执行的长尾，导致被分配处理这些值的执行线程会用更多的时间完成执行。

## 刷新 Schema（Schema Refresh）

OceanBase 数据库的 DDL 操作对系统对象的变更都发生在 Root Server 上，在 DDL 发生后 Root Server 会通知集群中的每一个节点最新的 Schema Version，节点在收到 Schema Version 后，和本地缓存的 Schema Version 比对，如果本地落后于全局版本，则从系统表中获取变更来更新本地缓存的系统对象信息，这个过程叫刷新 Schema。每个节点也会在后台定期对 Schema Version 对比，如落后也会自动触发刷新 Schema。

## 算子（Operator）

组成执行计划的基本单元。一般通过多个算子构成一棵执行树完成用户 SQL 请求。

### 1.21 T

## 统计信息（Statistics）

一个描述数据库中表和列信息的数据集合。在 OceanBase 数据库中，统计信息有表统计信息（table level statistics）和列统计信息（column level statistics）两种。

## 透传模式（Pass-through Mode）

是指在数据集成过程中，将数据直接传递给目标系统，而不进行转换或清洗处理。在透传模式下，数据只是简单地从源系统传输到目标系统，不进行任何数据转换、数据清洗或数据处理操作。

## TPC-C

TPC-C 是 TPC 组织（Transaction Processing Performance Council）推出的一系列性能测试标准中的一款，自推出以来，一直是数据库业界性能测试的重要参考。

## TPC-H

TPC-H 是一个业界常用的基于决策支持业务的 Benchmark，通过一系列在大量数据集上面执行的复杂查询请求，检验数据库系统的分析以及决策支持能力。

### 1.22 U

## Unit 调度 (Unit Scheduling)

对于每个 Zone，根据 Unit 的动态调度，达到均衡的策略：

- 属于同一个租户的若干个 Unit，会均匀分散在不同的 server 上。
- 属于同一个租户组的若干个 Unit，会尽量均匀分散在不同的 server 上。
- 当一个 Zone 内机器整体磁盘使用率超过一定阈值时，通过交换或迁移 Unit 降低磁盘水位线。
- 否则，根据 Unit 的 CPU 和内存规格，通过交换或迁移 Unit 降低 CPU 和内存的平均水位线。

## Unit Number

资源单元的个数。

## Unit Group

OceanBase 数据库 4.0 开始在租户管理上增加了限制，要求同一个租户所有 Zone 的 Unit 个数相同。系统为每个 Zone 的 Unit 进行了编号，相同编号的 Unit 组成一个 Unit Group。Unit Group 具有以下特性：

- 每个 Unit Group 分配唯一 ID，通过 `oceanbase.DBA_OB_UNITS` 视图的 `UNIT_GROUP_ID` 字段可以查看该 ID。
- 一个日志流只属于一个 Unit Group，并且只分布于该 Unit Group 的 Unit 上。因此 Unit Group 内所有 Unit 以日志流为单位，分布了相同的数据分区，从而框定了一组数据。同时，也就要求每个 Zone 的服务能力对等。
- OceanBase 数据库 4.0 不再支持按 Zone 个性化配置租户的 Unit 个数，只能按照 Unit Group 维度整体调整，比如需要为租户水平扩容资源，调大 Unit 个数，只能所有 Zone 统一扩容；相应的，租户缩容的场景，只能按 Unit Group 整体删除 Unit。通过 Unit Group 机制，保证了不同 Zone 上的数据分布是同构的。

## Universe

表示跨地域部署的所有 OceanBase 数据库。

## 1.23 W

## 微块 (Micro Block)

在宏块内部数据被组织为多个大小为 16KB 左右的变长数据块，称之为微块 (Micro Block)，微块中包含若干数据行 (Row)，微块是数据文件读 IO 的最小单位。每个数据微块在构建时都会根据用户指定的压缩算法进行压缩，因此宏块上存储的实际是压缩后的数据微



块，当数据微块从磁盘读取时，会在后台进行解压并将解压后的数据放入数据块缓存中。每个数据微块的大小在用户创建表时可以指定，默认 16KB，用户可以通过语句指定微块长度，但是不能超过宏块大小。

## 微块缓存（Block Cache）

微块在内存中的缓存，用于减少微块被频繁访问的 IO，提升查询性能。

## 微块索引缓存（Block Index Cache）

微块索引在内存中的缓存，用于提升频繁访问的查询性能。因为每个 SSTable 虽然以宏块组织，但是 2M 的粒度对于用户查询来说往往粒度太大，因此需要根据用户查询的范围在宏块中定位实际需要的微块，微块索引就是描述对每个宏块中所有的微块的范围，当需要访问某个宏块的微块时，需要提前装载这个宏块的微块索引，因为进行了前缀压缩，因此大小通常较小，并且在 OceanBase 数据库内部给予其较高优先级，因此一般命中率较高。

## 位置（Locality）

用来描述一个表的副本类型以及分布位置的方式。基本语法结构型为 `replicas@location`，由如下一些元素组成：

- 副本类型（replicas）：例如，F 表示全功能副本；L 表示日志型副本。
- 位置（location）：位置是系统已知的一组枚举值。位置是 Zone 的名称，如 hz1、bj2 等。
- 量词：不指定量词时，表示一个副本；用 `{n}` 表示 n 个副本。有一种特殊的量词 `{all_server}` 表示副本数和可用的 server 数相同。目前，由于一些实现上的考虑，一个分区在一个 Zone 中最多有一个全功能或日志型副本（这些类型的副本是 Paxos 复制组的成员），只读型副本在同一个 Zone 可以有多个。

## 位置信息缓存（Location Cache）

分区位置信息缓存。每个节点维护分区位置信息，在执行计划生成阶段，根据位置信息生成不同类型的执行计划；在执行阶段，根据位置信息将计划发送到相应的节点执行。位置信息缓存的更新是按需的，如果在语句执行过程中发生由于位置失效产生的错误，则刷新相应分区的位置信息。

## 物化视图（Materialized View）

物化视图是一种数据库对象，它与传统的视图不同，物化视图将查询结果作为数据实际存储于数据库中。

## 物化视图日志（Materialized View Log）



物化视图日志用于记录基表数据变更的日志。

## 无损切换（Switchover）

数据库主备库进行切换的一种切换方式。当数据库提供服务的主库出现故障或意外导致无法提供服务时，切换提供服务的集群为物理备库，该过程可逆。

## 无主选举（Election without leader）

分区没有 Leader 的情况下，这个分区的多副本进行的选主流程。无主选举的两个触发场景是集群重启分区第一次选举和分区原 Leader 故障，无主选举需要等原 Leader 的 Lease 过期后才能开始。

## 1.24 X

## XA 事务（XA Transaction）

XA 协议是由 X/Open 公司于 1991 年发布的一套标准协议。XA 是 eXtended Architecture 的缩写，因此该协议旨在解决如何在异构系统中保证全局事务的原子性。

## 系统变量（System Variables）

通过系统变量的设置使 OceanBase 数据库的行为符合您业务的要求。OceanBase 数据库的系统变量分为全局变量和 Session 变量。

## 系统配置项（System Parameters）

OceanBase 数据库的配置项分为集群级配置项和租户级配置项。通过配置项的设置可以控制整个集群的负载均衡、合并时间、合并方式、资源分配和模块开关等功能。

## 系统租户（sys Tenant）

系统租户是集群默认创建的租户，与集群生命期一致，负责管理集群和所有租户的生命周期。系统租户仅有一个 1 号日志流，只支持单点写入，不具备扩展能力。系统租户可以创建用户表，所有的用户表和系统表数据均由 1 号日志流服务。系统租户数据是集群私有的，不支持主备集群物理同步和物理备份恢复。

系统租户是应用系统访问 OceanBase 数据库的入口，客户端解析应用系统配置文件后访问 Config Server 获取系统租户的 IP 列表，然后访问系统租户逐级获取元数据，并最终建立与目标租户的连接。系统租户的容量是稳定性的一大挑战，大量应用系统同时重启时会产生建立连接的流量尖峰，短时间耗尽系统租户的工作线程，导致应用系统与目标租户建立连接失败。系统租户不具备横向扩展能力，可以对系统租户垂直扩容，或者调整集群配置项。

## 小版本冻结（Minor Freeze）

分区在内存中的增量数据超过阈值时，冻结该分区当前活跃 MemTable，不再接受新开启事务的写操作，新事务的写操作在该分区新的活跃 MemTable 中进行。

## 虚拟交换机（vSwitch）

指组成专有网络的基础网络设备。交换机可以连接不同的云资源。在专有网络内创建云资源时，必须指定云资源所连接的交换机。

## 1.25 Y

## 延时（Latency）

指计算机或网络系统处理任务或数据所需的时间，即从发送请求或任务开始到系统响应的时间间隔。

## 业务连续性（Business Continuity）

在业务遭遇各种内部或外部风险、威胁或灾害时，能够维持业务连续性和稳定性的能力和计划。

## 异地多活（Active Geo-redundancy）

异地多活（Active Geo-redundancy）是一种技术，旨在提高系统的可用性和灾难恢复能力，通过在多个地理位置之间复制和同步数据，确保数据的高可用性和一致性。异地多活通常涉及将主要的业务数据复制到多个数据中心或数据中心区域。

## 异地灾备（Geo-redundancy）

指在不同的地理位置上部署相同的计算设备、存储设备、网络设备等基础设施，以保证在其中一个地点发生灾难性故障时，另一个地点能够继续提供服务并保证数据安全可靠性。

## 用户租户（User Tenant）

与系统租户对应。用户租户是由用户创建的租户，对外提供完整的数据库功能，支持 MySQL 和 Oracle 两种兼容模式。用户租户支持服务能力水平扩展到多台机器上，支持动态扩容和缩容，内部会根据用户配置自动创建和删除日志流。用户租户的数据有更强的数据保护和可用性要求，支持跨集群物理同步和物理备份恢复，典型数据包括：Schema 数据、用户表数据、事务数据等。

## 远程计划（Remote Plan）

当执行计划只涉及到单表或分区表的单个分区，且该表或分区在其他节点时，该计划为"远程计划"。

## 远程执行 (Remote Execution)

接收用户请求和生成执行计划的数据库服务器和计划执行的数据库服务器不是同一个，并且只有一台数据库服务器执行该计划。

## 源端 (Source Connection Profile)

用于建立与数据源之间的连接的一组配置信息和参数。这些配置信息包括数据源的类型、主机名、端口号、用户名、密码、数据库名称等等。

## 优化器 (Optimizer)

优化器是决定用户查询执行计划的核心模块。通过访问相关数据的统计信息，结合 OceanBase 数据库内置的规则与代价模型，优化器为用户查询生成最佳的执行计划。

## 有主改选 (Re-election with leader)

分区有 Leader 的情况下，把分区 Leader 切换到指定 OBServer 的流程。有主改选不需要等原有 Leader Lease 过期。

## 预检查 (Pre-Check)

执行数据迁移操作之前对相关数据进行完整性、约束条件、关联关系以及权限等方面的检查，以确保操作的安全性和可靠性。

## 云服务器 (Cloud Server)

指虚拟的物理服务器，由服务商搭建维护，用户按需租赁使用。可在云端提供可靠的弹性计算服务，实现计算需求；随着业务需求的变化，实时扩展或缩减计算资源。

## 1.26 Z

## Zone

在 OceanBase 中，Zone 是指一个数据中心或一个物理区域，它是一个逻辑上的概念，通常包含多个存储节点，这些节点在物理上可以分布在不同的机房、不同的机架或不同的服务器上。一个 Zone 可以包含多个数据中心（比如机房），但是一个数据中心只能属于一个 Zone。

在 OceanBase 中，Zone 通常用于实现跨数据中心的数据容灾。OceanBase 会将数据按照一定的规则分布到不同的 Zone 中，从而实现数据的冗余备份。当一个 Zone 发生故障时，系统可以自动切换到备用 Zone 中的数据来保证数据的可用性。

除了数据容灾，OceanBase 的 Zone 还可以用于数据分片的容器。数据分片是一种将数据划分为多个分片，并将不同分片存储在不同的节点上的技术，可以提高系统的吞吐量和性能。

在 OceanBase 中，可以将不同的 Zone 作为不同的分片的 Primary Zone，从而实现分布式存储和处理数据。

## 增量合并（Incremental Compaction）

在 OceanBase 数据库的存储引擎中，宏块是 OceanBase 数据库基本的 IO 写入单位，在很多情况下，并不是所有的宏块都会被修改，当一个宏块没有增量修改时，合并可以直接重用这个数据宏块，OceanBase 数据库中将这种合并方式称之为增量合并。增量合并极大地减少了合并的工作量，是 OceanBase 数据库目前默认的合并算法。更进一步地，OceanBase 数据库会在宏块内部将数据拆分为更小的微块，很多情况下，也并不是所有的微块都会被修改，可以重用微块而不是重写微块。微块级增量合并进一步减少了合并的时间。

## 增量数据（Incremental Data）

执行增、删、改（insert、update、delete）等操作产生的修改数据，该部分数据尚未与基线数据合并，包括 MemTable 数据和转储数据。

## 增量数据（Change Data）

指在某个时间段内更新或新插入的数据。

## 增量数据表（MemTable）

内存中增量修改记录的集合。

## 增量同步（Change Synchronization）

是指在数据同步过程中，只传输发生变化的数据或文件，而不是传输全部的数据或文件。

## 只读可用区（Read Zone）

只读 Zone 是一种特殊的 Zone，在这个 Zone 里，只部署只读副本。通常当多数派副本故障的时候，OceanBase 数据库会停止服务，但在这种情况下，只读 Zone 能继续提供弱一致性读，即读 Zone、读库。这也是 OceanBase 数据库提供读写分离的一种方案。

## 正向切换（Forward Shifting）

指将数据从一个系统或平台迁移到另一个系统或平台的过程中，逐步将数据转移到新系统或平台的过程。

## 执行计划/计划（Execution Plan/Plan）

数据库中用以执行用户 SQL 请求的物理代码的集合，通常是一棵由算子构成的执行树。

## 执行计划缓存/计划缓存（Plan Cache）

执行计划在每台 server 上的缓存。SQL 语句的优化是一个比较耗时的过程，为了避免反复执行优化过程，生成的执行计划会加入到执行计划缓存中，以便再次执行该 SQL 时使用。每一个租户在每一台 server 上都有一个独立的执行计划缓存，用以缓存在此 server 上处理过的执行计划。

## 执行计划绑定（Execution Plan Binding）

用户通过 outline 绕过优化器而直接指定 SQL 的执行计划的过程，通常用于优化器生成的执行计划错误或者不够高效的场景。

## 执行计划匹配（Execution Plan Matching）

数据库为用户 SQL 选择在执行计划缓存中合适的执行计划的过程。

## 仲裁服务（Arbitration Service）

可以为 OceanBase 集群提供仲裁高可用能力，单个仲裁服务可以对接 N 个 OceanBase 集群，仲裁服务由仲裁 server 进程承载，目前单个仲裁服务只包含一个仲裁 server 进程。

## 仲裁成员（Arbitration Member）

仲裁成员和日志流副本是对等的，比如某个租户的部署方案为 2F+1A，那么该租户的所有日志流都有两个 F（全能型）副本和一个仲裁成员。

## 物理备库（Physical Standby Database）

物理备库是 OceanBase 数据库高可用解决方案的重要组成部分，当主库出现计划内或计划外（多数派副本故障）的不可用情况时，备库可以接管服务，并且提供无损切换（RPO = 0）和有损切换（RPO > 0）两种容灾能力，最大限度降低服务停机时间。

从 V4.1 版本开始，OceanBase 数据库支持租户级物理备库，用户可以为一个主租户创建一个或多个备租户，用户可以将资源密集型的报表操作分配到备集群，以便提高系统的性能和资源利用率。

## 主/从副本（Leader/Follower）

主/从副本定义了某一时刻某表分区副本的角色。对于 OceanBase 数据库，每个分区至少三副本，副本之间使用 Paxos 协议同步 Leader 的 Redo 到 Follower，策略上三个成员里只要有超过半数以上成员接收到 Redo 并落盘成功确认后，Leader 上的事务才可提交。每个分区的全部副本（三副本，五副本等）是一个独立的 Paxos Group，交付自行独立选主的能力，当一个 OceanBase 节点故障时，只有对应 OceanBase 数据库节点内部作为 Leader 的 observer 服务中断，OceanBase 集群服务会自动选出新的 Leader。



## 主租户（Primary Tenant）和备租户（Standby Tenant）

从 OceanBase V4.1 版本开始，租户有角色的概念（TENANT\_ROLE），一般分为两种角色：主租户（Primary Tenant）和备租户（Standby Tenant）。主租户可以对外提供完整的数据库服务能力，包括查询、DML、DDL等；备租户则仅提供灾备和只读服务的能力。备租户支持通过配置日志恢复源（log\_restore\_source）的方式来同步和恢复 Redo 日志，从而成为一个主租户的物理热备库。通过为一个主租户配置一个或多个备租户，可以实现租户级物理备库高可用解决方案。

## 主键（Rowkey）

与传统关系数据库的主键（Primary Key）相同，表中每一行数据的标识符，用于唯一标识表中的行，表中的数据按照 Rowkey 排序。

## 主可用区（Primary Zone）

Primary Zone 指的是分区的主副本所在的 Zone，可以为分区指定一个 Zone 的列表，当分区需要切主的时候，容灾策略会按照这个列表的顺序决定新主的偏好位置。

如果不设定 Primary Zone，系统会根据负载均衡的策略，在多个全功能副本里自动选择一个作为 Leader。

## 转储（Mini Compaction & Minor Compaction）

OceanBase 数据库中的转储可以理解为和其他 LSM-Tree 架构数据库的 Compaction 概念类似，主要负责 MemTable 刷盘转成 SSTable 以及多个 SSTable 之间的 Compaction 策略选择以及动作。OceanBase 数据库中采用的是 leveled 结合 size tiered 的 Compaction 策略，大致可以分为三层，其中 L1 和 L2 就是固定的 leveled 层次，L0 层是 size tiered，L0 内部还会继续根据写放大系数以及 SSTable 个数进行内部 Compaction 动作。

## 专有云网络（Virtual Private Cloud）

简称 VPC，是每个用户创建的自定义私有网络，不同的专有网络之间逻辑上彻底隔离，您可以在自己创建的专有网络内创建和管理云资源。

## 子计划（Sub Plan）

在分布式并行执行场景下，和 DFO 意思相同。

## 自适应游标共享（Adaptive Cursor Sharing）

一种可以让优化器每一个参数化 SQL 存储多个执行计划，并根据 SQL 语句中谓词的选择性来选择合适的计划的机制。

## 资源池（Resource Pool）

一个租户拥有若干个资源池，这些资源池的集合描述了这个租户所能使用的所有资源。一个资源池由具有相同资源规格（Unit Config）的若干个 Unit（资源单元）组成。一个资源池只能属于一个租户。每个 Unit 描述了位于一个 server 上的一组计算和存储资源，可以视为一个轻量级虚拟机，包括若干 CPU 资源、内存资源和磁盘资源等。

一个租户在同一个 server 上最多有一个 Unit。实际上，从概念上讲，副本是存储在 Unit 之中，Unit 是副本的容器。

## 资源单元（Unit）

租户在节点上的容器，描述租户在节点上的可用资源（CPU、MEMORY 等）。一个租户在一个节点只能同时存在一个 Unit。

## 资源管理计划（Resource Management Plan）

资源管理计划内容的容器，指定如何将资源分配给资源组。您可以通过激活特定的资源管理计划来控制资源的分配。一条资源管理计划可以对应多条资源管理计划内容。但一条资源管理计划中不能包含两条相同的资源管理计划内容。

## 资源管理计划内容（Resource Management Plan Config）

用于将资源组与资源管理计划相关联，并指定如何将资源分配给该资源组。

## 资源规格（Unit Config）

Unit Config 规定了一个 Unit 需要使用的计算存储资源（包含内存、CPU 和 IO 等）的规格，是一个资源配置信息。

## 资源组（Resource Group）

根据资源要求组合在一起的一组会话。系统将资源分配给资源组，而不是单个会话。

## 租户（Tenant）

OceanBase 数据库通过租户实现资源隔离，采用单集群多租户的管理模式。OceanBase 集群的一个租户相当于一个 MySQL 或者 Oracle 的实例。OceanBase 数据库的租户之间的资源和数据都是隔离的。租户拥有一组计算和存储资源，提供一套完整独立的数据库服务。



OceanBase 数据库上有系统租户和普通租户。系统租户下存放 OceanBase 数据库管理的各种内部元数据信息；普通租户下存放用户的各种数据和数据库元信息。

## 租户实例 (Tenant Instance)

为客户提供租户模式的 OceanBase 云数据库服务，当前支持 MySQL 模式。

## 组提交 (Group Commit)

组提交是为了优化写日志时的刷磁盘问题，将多个事务的日志聚在一起用一次 IO 完成持久化。

## 最大性能 (Maximum Performance)

这是默认的保护模式。它在最大限度地确保主集群性能的同时，还能保护用户的数据。在这种保护模式下，事务只需要等 REDO 日志在主集群持久化成功就可以立即提交。REDO 日志会异步的同步到备集群，但是不会阻塞主集群事务提交。因此，主集群的性能不会受备集群的同步延时影响。

## 最小化停机迁移 (Minimal Downtime Migration)

指在进行数据迁移的过程中，尽可能减少服务停机时间的一种策略。通常情况下，数据迁移或系统升级需要在一定时间内暂停服务以确保数据完整性和一致性。但是，停机时间过长可能会对业务造成重大影响，因此，最小化停机迁移的目标是尽可能缩短服务停机时间，减少对业务的影响。

## 仲裁服务

OceanBase 集群提供仲裁能力的一个独立进程。

## 仲裁 Server

部署仲裁服务的机器。

## 仲裁成员

开启了仲裁服务的租户，它的每个日志流在仲裁服务中对应的实例，称为仲裁成员。

## 仲裁副本

等价于仲裁成员。

## 仲裁降级

当半数全功能副本故障时，由仲裁服务参与投票的容灾行为，会将故障副本变更为 Learner（即不再参与日志同步）。

## 仲裁升级

当故障副本恢复时，由仲裁服务参与投票的恢复行为，会将故障副本重新变更为 Paxos Acceptor（即日志同步参与者）。