恒生电子股份有限公司

Oracle到LightDB迁移指南

LightDB

2021年8月

文档修改记录

版本	修订人	修订说明	批准人	发布日期
1.0.0	徐田原、杜方舟	初稿		2021/7/7
1.0.1	张君华、方俊	结构调整，内容删减、完善		2020/7/20

说　明

目录

1 引言

1.1 本文目的

Oracle是目前主流数据库之一，在各种业务系统中被广泛使用。本文提供了从Oracle迁移到LightDB的操作指南，包括语法特性、兼容性、操作指导等。

1.2 预期的读者和阅读建议

本文档预期读者为DBA和开发人员。

本文档提供了迁移指南和注意事项，DBA可根据本文档进行数据迁移的操作。

文档中如语法兼容性、数据类型、数据库函数、JDBC等方面的内容，可为业务开发人员提供帮助。

1.3 参考资料

1. 文中提到的函数可参考《SQL兼容性清单》.xlsx。
2. 更多信息可参见LightDB官网www.hs.net/lightdb。

2 为什么要迁移到LightDB

2.1 强大的功能及Oracle兼容性

LightDB依托恒生强大的业务经验、研发能力及开源社区，使其在各方面与Oracle的体验接近甚至在一些方面有所超越。Oracle需要更复杂的持续管理，因为所有数据库配置必须与数据模式和自定义代码共同发展。极端的复杂性还增加了产生错误的可能性，最终可能导致严重的错误，这些错误需要更多地时间和金钱来解决。这就导致在Oracle中通常是将每年更改次数限制为两次。

LightDB在基础功能上中引入了许多突破性功能，这些功能使其成为与Oracle真正的竞争者，例如分区增强、并行查询和逻辑复制。

LightDB 的可以使用函数和条件索引，这使得LightDB数据库的调优非常灵活，MySQLl就没有这个功能。条件索引在web应用中很重要。

LightDB有极其强悍的 SQL 编程能力，有非常丰富的统计函数和统计语法支持，比如窗口函数。

	LightDB	Oracle
兼容性	LightSQL PL/pgSQL与其他关系数据库（例如Oracle）兼容，这使得向LightSQL迁移变得相对容易	Oracle基础架构不提供开源RDBMS的兼容性扩展
扩展组件	LightSQL提供了许多扩展组件，包括：pg_fincore模块将数据库表，索引CACHE到OS层面的缓存里，只要内存足够大，可以将需要的数据都CACHE到OS 内存中，这极大的提高了应用的处理速度;pg_buffercache模块提供了一种实时检测共享缓冲区的方法等	Oracle提供了商业附加组件，需要额外的许可费用
调优	LightSQL提供了更轻量级的调优功能，例如Automatic Tuning Optimizer，而LightDB EM平台提供了高级的慢查询分析	由于数百种调优变量和复杂的系统要求，Oracle需要大量的安装和配置工作。自动工作量存储库（AWR）和数据库顾问提供的大多数调优功能都与需要企业版的Oracle Enterprise Manager、Database/Grid control捆绑在一起
支持的操作	Linux	FreeBSD,HP-UX,Linux,NetBSD,OpenBSD,OS

系统		X,SolarisUnix,Windows
客户端支持的语言	C#,C++, Delphi, Nodejs,Java,JavaScript, Perl,PHP,Python	.Net,C++,Delphi,Java,JavaScript (Node.js),Perl,PHP,Python
支持帮助	可以从LightDB官网获得LightDB技术支持和增强服务	可以从Oracle 官网获得技术支持和增强服务

2.2 更低的总拥有成本（TCO）

总拥有成本（TCO）是目前广泛采用的技术成本评价标准，它包括购买成本和每年的使用成本。Oracle将许多功能分解为单独的许可证，因此您必须为这些额外的功能进行支付，而且许多高级功能仅在其企业版中可用，例如滚动升级、高级安全性、高级复制和分区。 在此分类中。LightDB在提供相同功能和服务的前提下，其购买和使用成本大大低于Oracle。

2.3 原生HTAP

LightDB发布的HTAP版本，可同时支持在线事务和数据分析，不需为每种业务类型安装部署不同的数据库，也无需在数据库之间同步数据。在同一份数据上同时支撑OLTP和OLAP场景，避免了传统架构中在线与离线数据库之间大量的数据交互。同时支持弹性扩容，轻松应对高并发、海量数据场景。

2.4 适配国产信创

除了支持广泛使用的国外操作系统如RHEL、CentOS，LightDB把重点放在对国产软硬件平台的支持，CPU如ARM处理器、海光处理器，操作系统如麒麟OS、openEuler。

2.5 7*24原厂服务

恒生电子专业的售前、售后工程师团队可提供7*24小时不间断的技术服务。如果在使用LightDB时遇到任何问题，可通过智能客服、工单、热线电话等方式寻求帮助。我们将竭诚为您服务。

3 LightDB vs. Oracle基础特性比较

3.1 术语差异

LightDB	Oracle
集群（Cluster）	实例（Instance）
WAL	redo
元组（tuple）	行
关系	表
属性	列
页	块

3.2 LightDB vs. Oracle 架构对比

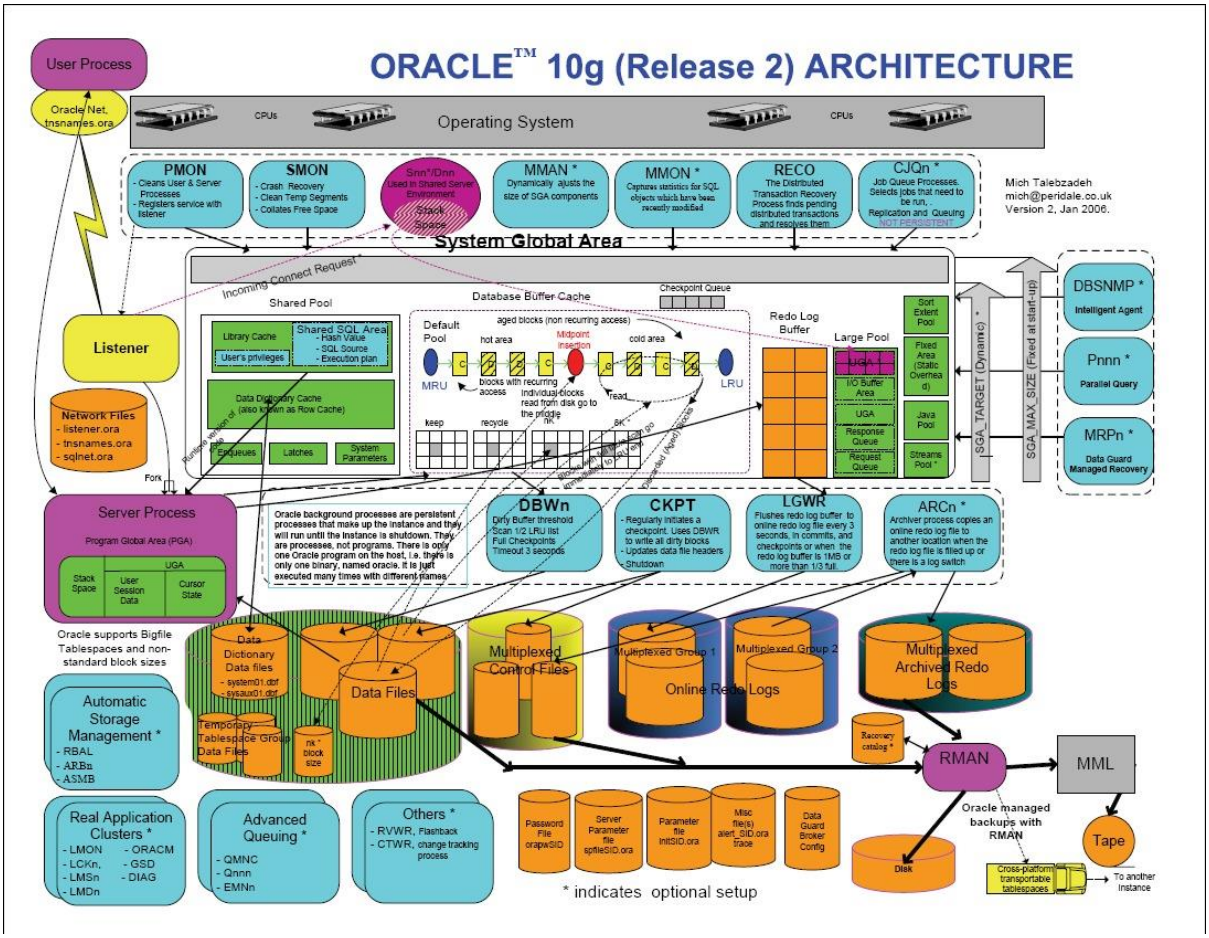

图 Oracle体系架构

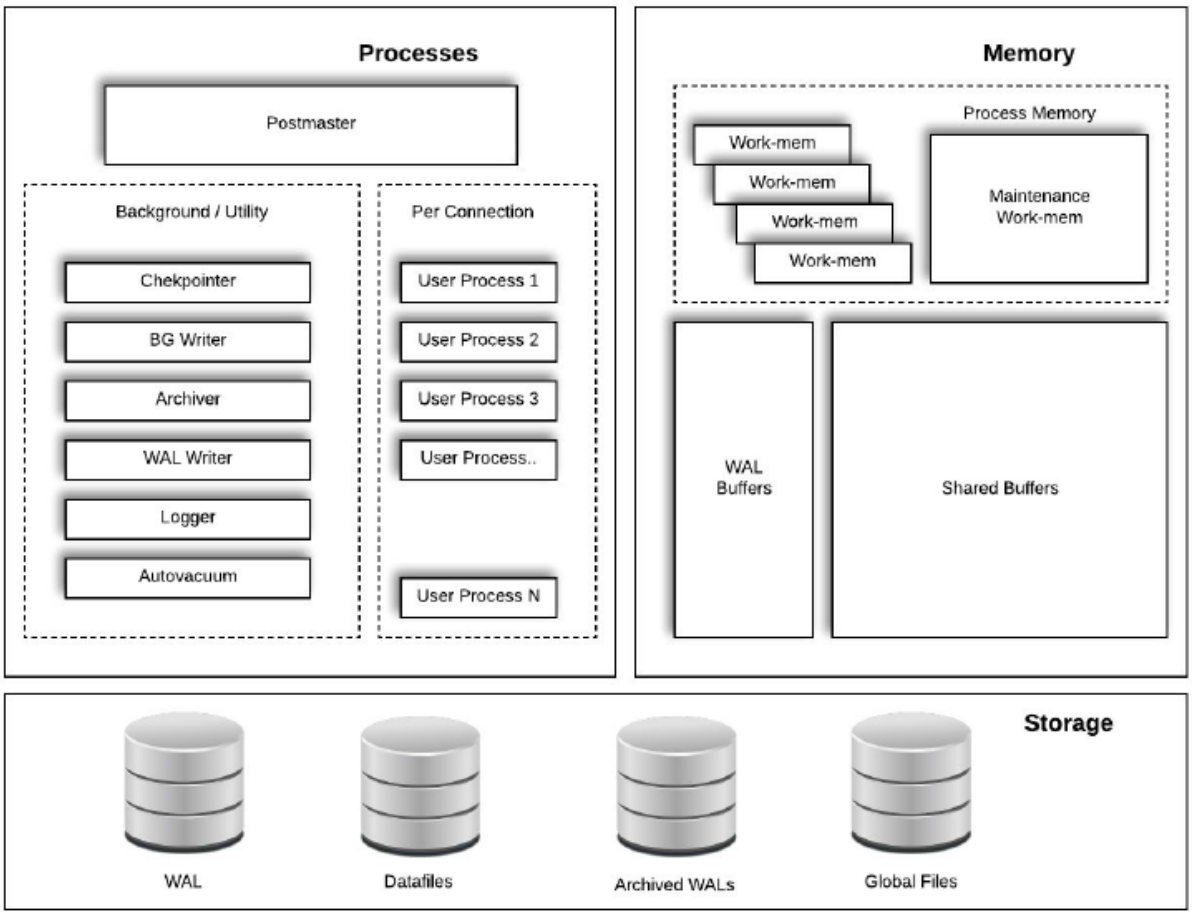

图 LightDB体系架构

LightDB和Oracle一样，都是采用多进程架构。不同的是LightDB没有监听器的概念，由postmaster负责监听。除了工作进程，LightDB由若干辅助进程如logger、checkpointer、walwriter、background writer、autovacumm；当启用集群复制功能时，还会为每一个连接创建walreceiver/walsender进程。

LightDB与Oracle的关键组件对应关系如下表。

Oracle	LightDB	说明
SGA	shared_buffers	共享缓存
PGA	work_mem	进程私有内存
PMON	postmaster	
TNS Listener	postmaster	

3.3 LightDB vs. Oracle 容量规格

特性	Oracle	LightDB
数据库大小	不限	不限
关系大小	32TB 基于8192字节块大小	32 TB 基于块大小缺省值为8192字节
每个表的记录数	不限	受限于可以放到4,294,967,295个页中的元组数
每个表的列数	1000	1600 同时受限于元组大小，需要能放到单个页中
字段大小	(4 GB - 1) * DB_BLOCK_SIZE （BLOB类型）	1 GB
标识符长度	30	63字节 可以通过重新编译lightDB增大
每个表的索引数	不限	不限 受每个数据库最大关系数的约束
每个索引的列数	32	32 可以通过重新编译LightDB增大
分区键数量	16	32 可以通过重新编译LightDB增大

由于要存储的元组必须适合单个8192字节的堆页面，因此表的最大列数进一步减少。例如，除元组头外，由1600个int列组成的元组将占用6400字节并可以存储在堆页面中， 而一个包含1600个bigint列的元组将消耗12800字节，因此无法放入单个堆页面。 当类型为比如text，varchar和char的可变长度字段时， 如果值的长度足够大，它们的值可以存储在行外的TOAST表中。 表堆中的元组中只保留18个字节的指针。对于较短长度的可变长度字段，使用4字节或1字节的字段头，并且该值存储在堆元组内部。

3.4 常用字符集

名称	描述	语言	是否服务器端?	字节/字符	别名
BIG5	Big Five	繁体中文	否	1-2	WIN950, Windows950
EUC_CN	扩展UNIX编码-中国	简体中文	是	1-3	
EUC_JP	扩展UNIX编码-日本	日文	是	1-3	
EUC_JIS_2004	扩展UNIX编码-日本, JIS X 0213	日文	是	1-3	
EUC_KR	扩展UNIX编码-韩国	韩文	是	1-3	
EUC_TW	扩展UNIX编码-台湾	繁体中文，台湾话	是	1-3	
GB18030	国家标准	中文	否	1-4	
GBK	扩展国家标准	简体中文	否	1-2	WIN936, Windows936
ISO_8859_5	ISO 8859-5, ECMA 113	拉丁语/西里尔语	是	1	
ISO_8859_6	ISO 8859-6, ECMA 114	拉丁语/阿拉伯语	是	1	
ISO_8859_7	ISO 8859-7, ECMA 118	拉丁语/希腊语	是	1	
ISO_8859_8	ISO 8859-8, ECMA 121	拉丁语/希伯来语	是	1	
JOHAB	JOHAB	韩语	否	1-3	
KOI8R	KOI8-R	西里尔语（俄语）	是	1	KOI8
KOI8U	KOI8-U	西里尔语（乌克兰语）	是	1	

LATIN1	ISO 8859-1, ECMA 94	西欧	是	1	ISO88591
LATIN2	ISO 8859-2, ECMA 94	中欧	是	1	ISO88592
LATIN3	ISO 8859-3, ECMA 94	南欧	是	1	ISO88593
LATIN4	ISO 8859-4, ECMA 94	北欧	是	1	ISO88594
LATIN5	ISO 8859-9, ECMA 128	土耳其语	是	1	ISO88599
LATIN6	ISO 8859-10, ECMA 144	日耳曼语	是	1	ISO885910
LATIN7	ISO 8859-13	波罗的海	是	1	ISO885913
LATIN8	ISO 8859-14	凯尔特语	是	1	ISO885914
LATIN9	ISO 8859-15	带欧罗巴和口音的LATIN1	是	1	ISO885915
LATIN10	ISO 8859-16, ASRO SR 14111	罗马尼亚语	是	1	ISO885916
MULE_INTERNAL	Mule内部编码	多语种编辑器	是	1-4	
SJIS	Shift JIS	日语	否	1-2	Mskanji, ShiftJIS, WIN932, Windows932
SHIFT_JIS_2004	Shift JIS, JIS X 0213	日语	否	1-2	
SQL_ASCII	未指定（见文本）	任意	是	1	
UHC	统一韩语编码	韩语	否	1-2	WIN949, Windows949
UTF8（推荐，LightDB默认字符集）	Unicode, 8-bit	所有	是	1-4	Unicode
WIN866	Windows CP866	西里尔语	是	1	ALT
WIN874	Windows CP874	泰语	是	1	
WIN1250	Windows CP1250	中欧	是	1	

WIN1251	Windows CP1251	西里尔语	是	1	WIN
WIN1252	Windows CP1252	西欧	是	1	
WIN1253	Windows CP1253	希腊语	是	1	
WIN1254	Windows CP1254	土耳其语	是	1	
WIN1255	Windows CP1255	希伯来语	是	1	
WIN1256	Windows CP1256	阿拉伯语	是	1	
WIN1257	Windows CP1257	波罗的海	是	1	
WIN1258	Windows CP1258	越南语	是	1	ABC, TCVN, TCVN5712, VSCII

4 LightDB vs. Oracle 结构比较

4.1 数据库（database）、模式（schema）与用户（user）

在Oracle 12c之前，实例和数据库通常是一一对应的（Oracle RAC多个实例对应一个数据库）。在Oracle 12c+的可插拔版本中，一个实例可以包含多个数据库，会话只能连接到一个数据库，数据库之间的对象不能同时访问或关联。数据库内可以包含多个schema，schema和用户的概念和实现都代表同一个意思。

在LightDB中，实例可以包含多个数据库，会话只能连接到一个数据库，数据库之间的对象不能同时访问或关联。数据库内可以包含多个schema，默认为public，schema和用户不是相同的概念，schema相当于namespace，用户是真正用来登录的。

LightDB的组织层次如下图所示：

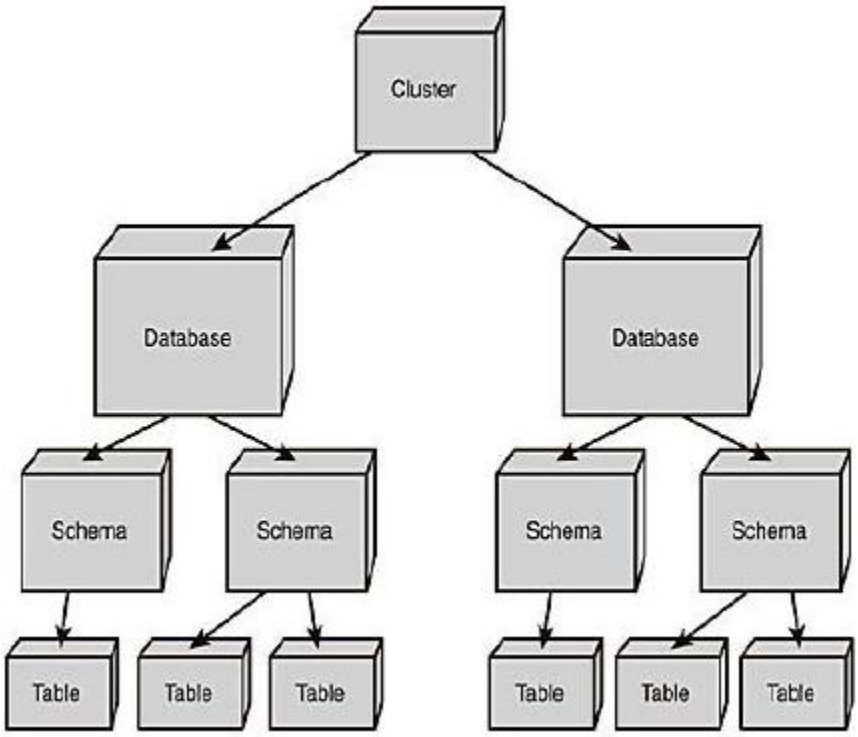

图 LightDB组织层次

4.2 表空间

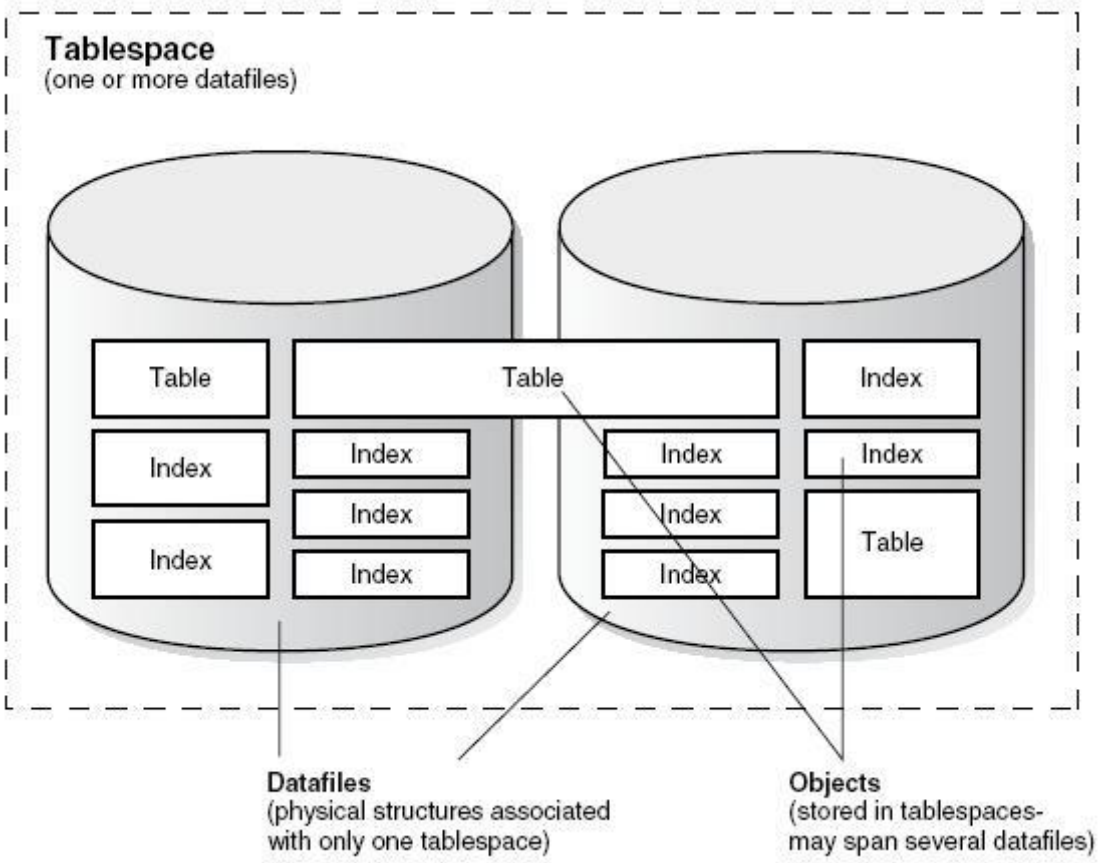

图 Oracle表空间

Oracle的表空间可以关联到多个数据文件；LightDB的表空间与一个指定目录绑定，数据文件只能创建在对应目录中。另外Oracle可以为undo日志使用独立的表空间。

LightDB没有Datafile这一层，创建表或索引时直接指定表空间。关联的对象（如表和索引）可以指定不同的表空间。

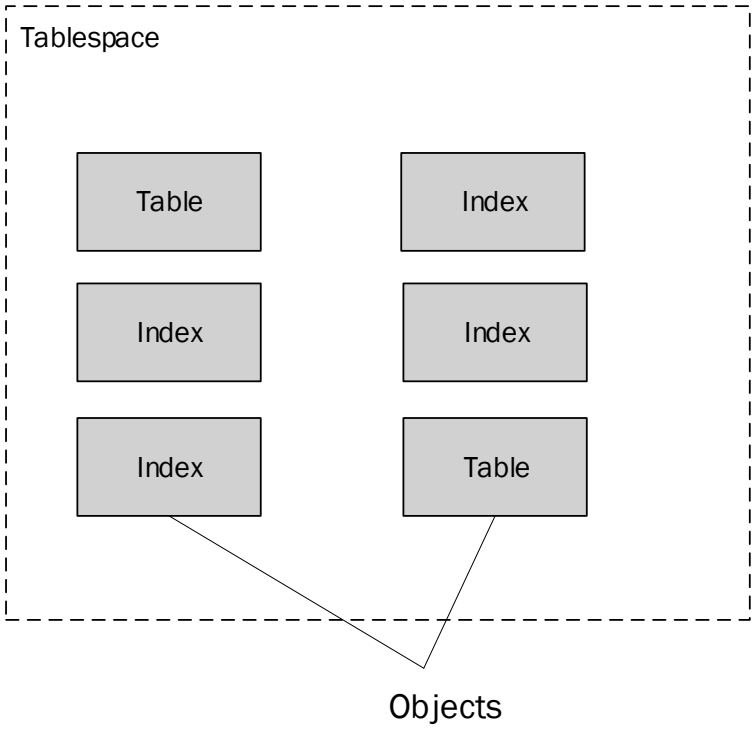

图 LightDB表空间

4.3 数据类型映射

Oracle	LightDB
varchar、varchar2、nvarchar、nvarchar2	char, varchar, text
char，nchar	char, varchar, text
clob，long	varchar,text,jsonb
number	numeric
blob,raw,long_raw	bytea 如果大对象是 json 可以换做 jsonb
date	date or timestamp Timestamp with timezone
date +\-	date + inteval ‘ N day/minute’
nls_date_format	To_char to_date
TIMESTAMP	date,timestamp, timestamptz, char,varchar, text
TIMESTAMP WITH TIME ZONE	date, timestamp, timestamptz, char, varchar, text
TIMESTAMP WITH LOCAL TIME ZONE	date, timestamp, timestamptz, char, varchar, text

INTERVAL YEAR TO MONTH	interval, char, varchar, text
INTERVAL DAY TO SECOND	interval, char, varchar, text
ROWID	CTID
XMLTYPE	XMLTYPE
使用带有json check约束的 VARCHAR2, CLOB, BLOB 类型	JSON 和快速二进制JSONB
sequence 序列	serial/bigserial（此外，LightDB未来还将支持使用雪花ID）
UUID-使用sys_guid()函数	使用uuid-ossp插件
大对象- BLOB/CLOB	bytea/text

注1：MySQL和LightDB字段映射可参见https://dev.mysql.com/doc/workbench/en/wb-migration-database-postgresql-typemapping.html。

注2：Oracle到LightDB的迁移，及使用开发工具反向生成DDL可参见12 Oracle到LightDB迁移工具。

注3：完整的数据类型映射参见《数据库无关性SQL清单》.xlsx。

4.3.1 LightDB独有的数据类型转换语法

同其它数据一样，LightDB支持函数调用式的类型转换（如to_char、to_date、to_number）和CAST。另外它也支持独有的转换语法。

LightDB支持显示类型转换语法expression::type：有时隐式转换也许不是想要的，或不够明确，可以使用显示类型转换。

```
test=# select 'Hello'::text;
 text
-------
 Hello

test=# select 12345::text;
 text
-------
 12345
```

这种语法兼容性不高，所以建议使用如to_char这样函数式的类型转换。使用示例如下，具体说明可参考LightDB文档。

```
to_char(timestamp '2002-04-20 17:31:12.66', 'HH12:MI:SS') → 05:31:12
to_char(interval '15h 2m 12s', 'HH24:MI:SS') → 15:02:12
```

```
to_char(125.8::real, '999D9') → 125.8
-- D表示小数点位置，S表示符号位置，9表示该位置是一个数字
to_char(-125.8, '999D99S') → 125.80-
to_date('05 Dec 2000', 'DD Mon YYYY') → 2000-12-05
-- G表示数字分隔符（与语言区域相关，这里是逗号）
to_number('12,454.8-', '99G999D9S') → -12454.8
to_timestamp('05 Dec 2000', 'DD Mon YYYY') → 2000-12-05 00:00:00-05
```

4.3.2 字符串拼接

LightDB兼容Oracle字符串拼接语法：

```
test=# select 'Light' || 'DB' as name;
  name
---------
 LightDB
```

4.4 表和分区

表和分区特性	Oracle	LightDB
临时表	全局临时表和本地临时表。 数据支持会话级的临时表和事务级的临时表	支持（使用 LOCAL TEMP 表）
视图	支持	支持
创建表-INITTRANS, MAXEXTENTS（存储子句）	支持，INITTRANS 参数表示高并发时事务会锁住同一个对象，oracle 会使用块的一部分空间来保存哪些事务将哪些块中元素锁定，这个空间的大小由 INITTRANS 来决定,默认为 2,事务表会根据需要动态扩展，最大达到 MAXTRANS 个条目（假设块上有足够的自由空间）。	LightDB 创建表需要删掉这些参数
创建表-PCTFREE	支持,Oracle 中 PCTFREE 这个参数定义了一个块保留空间的百分比，保留空间是为了将来可能发生的更新操作，因为更新可能增大被更新行占用的空间，如果此时该块没有可利用空间，就会发生航迁移，从而会降低 I/O 性能	LighDB 需要使用填充因子，with (fillfactor=80)语句
分区	支持(按范围,哈希，列表分区)	支持（按范围,哈希，列表分区）
子分区	支持	支持

间隔分区	支持	不支持
全局索引	支持	不支持
表空间	支持	支持

4.4.1 数据组织方式

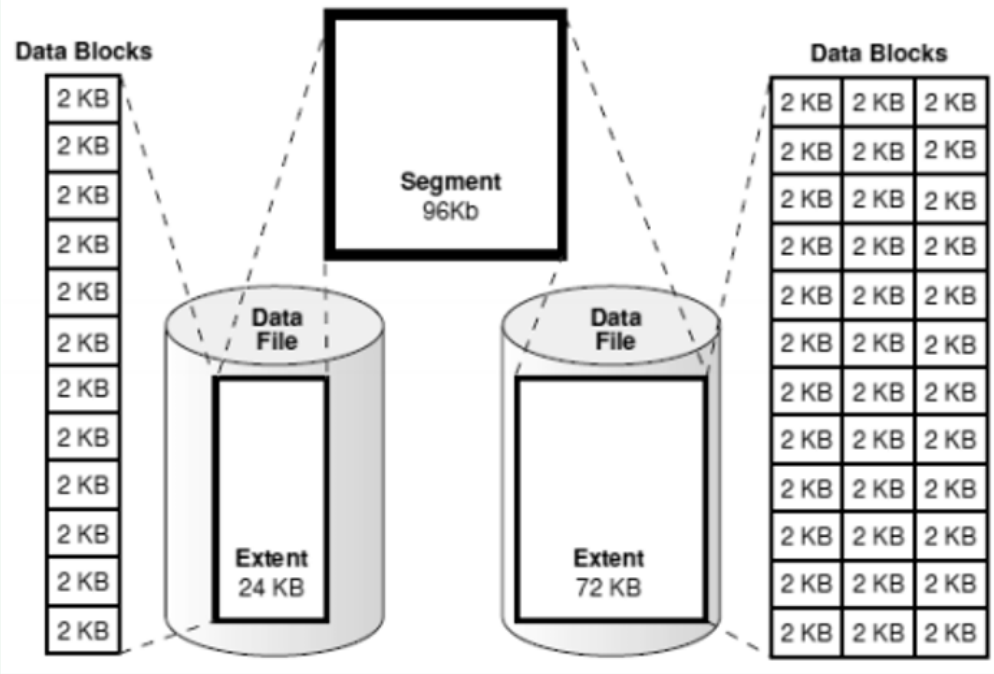

图 Oracle 段、区段、数据文件、数据库组成示意图

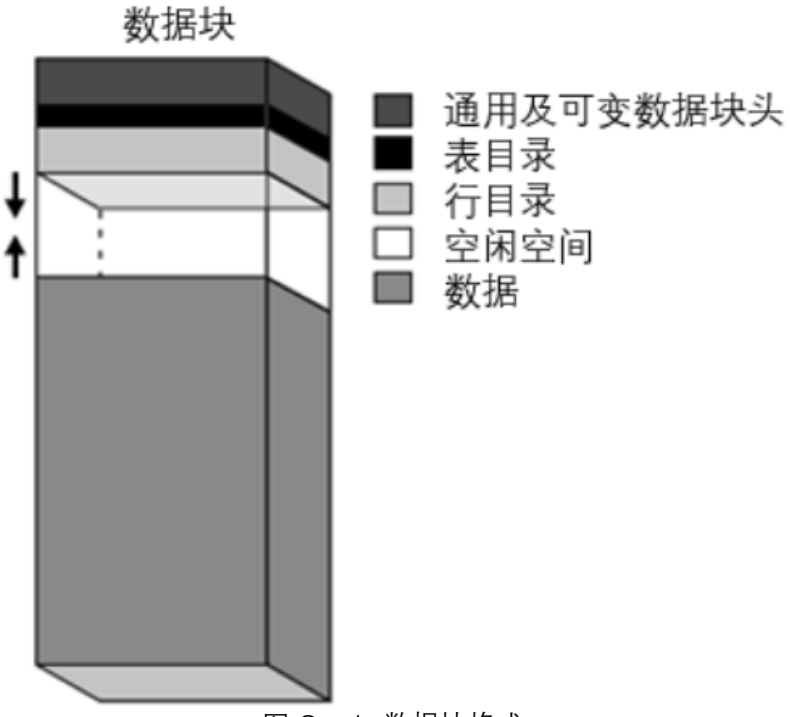

图 Oracle 数据块格式

一个表空间由很多个数据文件组成，每个数据文件包含很多个逻辑区段，物理上

由很多数据块组成。数据块是 Oracle 能够和使用的最小逻辑单元，一个 Oracle数据块通常由多个文件系统块组成。

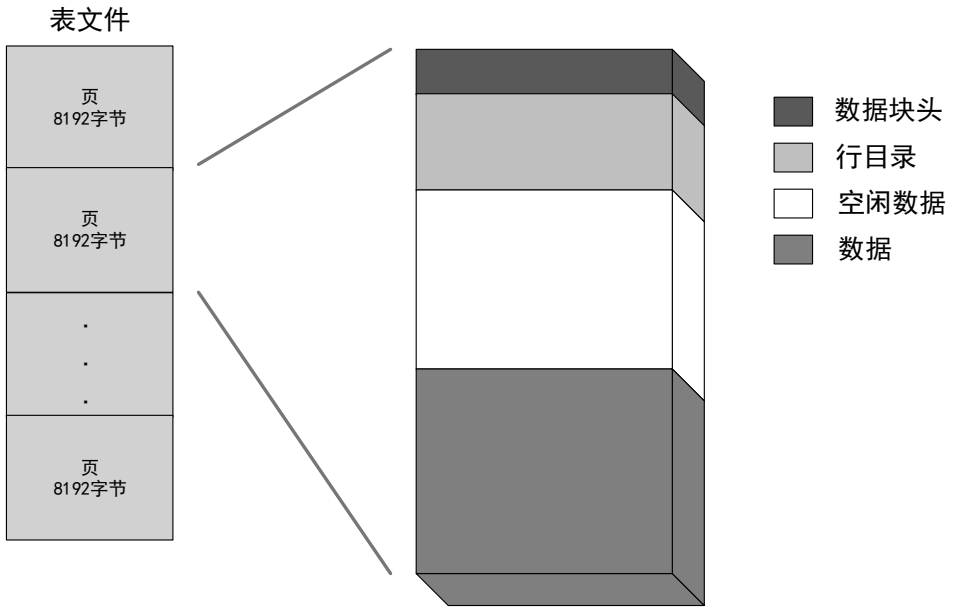

图 LightDB数据块组织方式

LightDB数据文件内部被划分成固定的区块，称为页。一页默认大小为8KB。每一页有自己的页头数据，页头后面紧跟行指针。与Oracle数据块类似，LightDB页内行指针往页尾生长，行数据从页尾往页头生长。

4.4.2 LightDB分区语法及用例

LightDB分区是把逻辑上的一个大表分割成物理上的子表。LightDB提供了一种方法，如何把一个表划分成称为分区的片段。被划分的表被称作分区表。分区不仅能带来访问速度的提升，关键的是，它能带来管理和维护上的方便。

主表/父表/Master Table：该表是创建子表的模板。它是一个正常的普通表，但正常情况下它并不储存任何数据。

子表/分区表/Child Table/Partition Table：这些表继承并属于一个主表。子表中存储所有的数据。主表与分区表属于一对多的关系，也就是说，一个主表包含多个分区表，而一个分区表只从属于一个主表。

语法格式如下：

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF
NOT EXISTS ] table_name
OF type_name [ (
{ column_name [ WITH OPTIONS ] [ column_constraint [ ... ] ]
| table_constraint }
[, ... ]
```

```
) ]
[ PARTITION BY { RANGE | LIST | HASH } ( { column_name | ( expression ) }
[ COLLATE collation ] [ opclass ] [, ... ] ) ]
[ USING method ]
[ WITH ( storage_parameter [= value] [, ... ] ) | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace_name ]
```

举例如下：

```
lightdb =# CREATE TABLE measurement (
    city_id         int not null,
    logdate         date not null,
    peaktemp        int,
    unitsales       int
) PARTITION BY RANGE (logdate);
CREATE TABLE
```

4.4.3 临时表

Oracle8i开始支持全局临时表，Oracle18c开始支持私有临时表。临时表的使用很大程度上降低了redo的负担。LightDB也有类似的概念及实现。

全局临时表Oracle示例：

```
CREATE GLOBAL TEMPORARY TABLE my_temp_table (
  id           NUMBER,
  description  VARCHAR2(20)
)
ON COMMIT PRESERVE ROWS;
```

全局临时表LightDB示例：

```
CREATE GLOBAL TEMPORARY TABLE my_temp_table (
id number,
description varchar(20)
)
ON COMMIT PRESERVE ROWS;
```

Oracle私有临时表的表名要以PRIVATE_TEMP_TABLE_PREFIX（默认值为"ORA$PTT_"）开头，否则建表失败：

```
CREATE PRIVATE TEMPORARY TABLE ora$ptt_my_temp_table (
  id           NUMBER,
```

```
  description  VARCHAR2(20)
)
ON COMMIT DROP DEFINITION;
```

为了兼容性，LightDB有LOCAL关键字，没有实际的效果。所以建临时表时建议不使用GLOBAL/LOCAL修饰。

4.5 主键ID

4.5.1 serial语法

LightDB中，smallserial、serial和bigserial类型不是真正的类型，它们只是为了创建唯一标识符列而存在的方便符号，类似Oracle 序列作为默认值的语法。

举例说明，下面一个语句创建了一个整数列并且把它的缺省值安排为从一个序列取值：

```
CREATE TABLE tablename (
    colname SERIAL
);
```

等价于以下语句：

```
CREATE SEQUENCE tablename_colname_seq AS integer;
CREATE TABLE tablename (
    colname integer NOT NULL DEFAULT nextval('tablename_colname_seq')
);

ALTER SEQUENCE tablename_colname_seq OWNED BY tablename.colname;
```

注：LightDB 在后续版本将支持自增语法。

4.5.2 UUID

Oracle中UUID是内建类型，用法示例如下：

CREATE TABLE myTable (id STRING AS UUID GENERATED BY DEFAULT, name STRING, PRIMARY KEY (id));

LightDB 需要使用uuid-ossp插件来支持uuid。这个插件提供了一套生成uuid的函数。示例如下。不支持在建表时使用uuid类型。

```
lightdb =# create extension "uuid-ossp";
 CREATE EXTENSION
lightdb=# create or replace function sys_guid() returns uuid as $$ select
```

```
uuid_generate_v4();
$$ language sql strict; CREATE FUNCTION
lightdb =# select sys_guid();
sys_guid
------------------------------------------------------------------
92bbbf05-a23c-41b3-95d4-8732c93d95dd
 (1 row)

lightdb =# select sys_guid();
sys_guid
------------------------------------------------------------------
37e34cfb-46aa-44ed-9403-9e23b6c2bfc0
 (1 row)
```

4.6 索引

索引是提高数据库性能的常用途径。比起没有索引，使用索引可以让数据库服务器更快找到并获取特定行。但是索引同时也会增加数据库系统的日常管理负担，因此我们应该聪明地使用索引。

LightDB提供了多种索引类型： B-tree、Hash、GiST、GIN 和 BRIN。每一种索引类型使用了 一种不同的算法来适应不同类型的查询。大多数情况下，我们使用比较常用的B-tree索引。

B-tree索引适合处理那么些能够按照顺序存储的数据，它适合用于比较操作时经常用到的字段。

Hash索引只能处理简单的等于比较。当一个字段涉及到使用“=”操作符进行比较时查询规划器会考虑使用Hash索引。

GiST（Generalized Search Tree）和GIN（Generalized Inverted Index）索引可用于全文搜索（full text search）。

GiST索引不只是一种索引，还是一种架构，可以实现很多不同的索引策略。所以，GiST索引可以使用的特定操作符类型高度依赖于索引策略(操作符类)。

GIN索引是反转索引，它可以处理包含多个键的值(比如数组)。和GiST索引类似，GIN支持用户定义的索引策略，GIN索引可以使用的特定操作符类型根据索引策略不同而不同。

BRIN索引（Block Range INdexes）存储关于存储在表的连续物理块范围内的值

的摘要。像GiST, GIN一样，BRIN可以支持许多不同的索引策略，并且可以使用BRIN索引的特定操作符根据索引策略而变化。对于具有线性排序顺序的数据类型，索引数据对应于每个块范围的列中值的最小值和最大值。BRIN适用于数据变动不大的列，相比B-tree索引，BRIN索引只需要非常小的空间，达到近似的搜索速度。

Oracle与LightDB对索引的支持对比如下：

索引类型	Oracle	LightDB
B-Tree 索引	支持	支持
Hash 索引	支持	支持
Expressions 表达式索引	支持	支持
Partial 局部索引	支持	支持
Reverse 反向索引	支持	支持 使用函数索引可用于反转字段的顺序
Bitmap 位图索引	支持	使用 GIN 索引
Block Range Index BRIN 索引	支持	支持
GIST 索引: Gist(Generalized Search Tree)，即通用搜索树，轻松创建专用索引	不支持	支持
GIN 索引（倒排索引来源于搜索引擎的技术，可以说是搜索引擎的基石）	不支持	支持
Full Text Search （全文搜索）	支持	支持

4.6.1 LightDB几种索引使用场景及示例

（1）B-tree适合所有的数据类型，支持排序，支持大于、小于、等于、大于或等于、小于或等于的搜索。

```
lightdb=# create table t_btree(id int, info text);
CREATE TABLE
lightdb=# insert into t_btree select generate_series(1,10000),
md5(random()::text) ;
INSERT 0 10000
lightdb=# create index idx_t_btree_1 on t_btree using btree (id);
CREATE INDEX
```

（2）hash索引存储的是被索引字段VALUE的哈希值，只支持等值查询。

```
lightdb=# create table t_hash (id int, info text);
CREATE TABLE
lightdb=# insert into t_hash select generate_series(1,100),
repeat(md5(random()::text),10000);
INSERT 0 100
lightdb=# create index idx_t_hash_1 on t_hash using hash (info);
```

（3）GiST索引适用于多维数据类型和集合数据类型，和B-tree索引类似，同样适用于其他的数据类型。和B-tree索引相比，GiST多字段索引在查询条件中包含索引字段的任何子集都会使用索引扫描，而B-tree索引只有查询条件包含第一个索引字段才会使用索引扫描。缺点是GiST索引创建耗时较长，占用空间也比较大。创建GiST索引的前提是已经编译并安装了GiST的扩展。

```
lightdb=# create table tbl_index(a bigint,b timestamp without time zone,c
varchar(12));
CREATE TABLE
lightdb=# insert into tbl_index (a,b,c)  select
generate_series(1,3000000),clock_timestamp()::timestamp(0) without time
zone,'got u';
INSERT 0 3000000
lightdb=# create extension btree_gist;
CREATE EXTENSION;
lightdb=# create index idx_gist_tbl_index_a_b on tbl_index using gist(a,b);
CREATE INDEX;
```

（4）GIN(Generalized Inverted Index, 通用倒排索引)是一个存储对(key, posting list)集合的索引结构，其中key是一个键值，而posting list 是一组出现过key的位置。如('hello', '14:2 23:4')中，表示hello在14:2和23:4这两个位置出现过，在LightDB中这些位置实际上就是元组的tid。

在表中的每一个属性，在建立索引时，都可能会被解析为多个键值，所以同一个元组的tid可能会出现在多个key的posting list中。通过这种索引结构可以快速的查找到包含指定关键字的元组，因此GIN索引特别适用于支持全文搜索，而GIN索引模块也就是为了支持全文搜索而开发的。使用gin索引需要增加pg_trgm拓展。

```
lightdb=# create table users (id bigint,b timestamp without time
zone,username text);
CREATE TABLE;
lightdb=#CREATE EXTENSION pg_trgm;
CREATE EXTENSION;
lightdb=#CREATE INDEX trgm_idx_users_username ON users USING gin (username
gin_trgm_ops);
```

```
CREATE INDEX;
```

5 LightDB vs. Oracle常用函数

详细对比及数据库无关函数参见《SQL兼容性清单》。

6 LightDB vs. Oracle开发特性

数据库 / 特性	Oracle	LightDB
WITH 表达式	支持	支持，可以作为通用表表达式，也可以作为临时表使用
序列	支持	支持，直接在表中指定字段类型为serial或bigserial类型，也可以先创建序列名称，然后在新建的表中列属性指定序列就可以了，该列需int 类型，详情见附录A
表	支持的表类型： ● 堆表（默认） ● IOT 表	支持的表类型 堆表（默认）
存储过程内返回游标	出参中显示定义游标	需要声明setof record返回类型或声明refcursor返回类型
主键	支持	支持
分页查询（为了保证分页性能稳定性，通常建议使用定位串）	10g 以及之前的版本使用基于rownum的两层嵌套循环 12c 以及之后的版本支持offset [m] rows fetch next [n] rows only	limit 子句
优化器提示	通过 select /*+ */语句级别控制，可控制连接方式、索引、基数、并行等任何方面，使用错误不影响逻辑和编译	通过 select /*+ */语句来控制连接顺序、索引、基数等方面，保证sql最优执行，使用错误不影响逻辑和编译
对象名大小写	不敏感 一般对象名均使用大写	Linux/unix 下敏感，尽可能全部小写 内部的数据字典均使用小写 如要查询或调用大写的对象，需在对应的名称上加双引号，如"NODE"
事务隔离级别	提供了SQL-92标准中的READ COMMITTED (读已提交)和SERIALIZABLE (可串行化)，同时提供了非SQL-92标准的READ-ONLY(读一致性)	READ COMMITTED REPEATABLE READ SERIALIZABLE
锁	ROW SHARE ROW EXCLUSIVE SHARE UPDATE SHARE SHARE ROW EXCLUSIVE	ACCESS SHARE ROW SHARE ROW EXCLUSIVE SHARE UPDATE EXLUSIVE SHARE

	EXCLUSIVE	SHARE ROW EXCLUSIVE EXCLUSIVE
创建数据库	DBCA	initdb
层次查询	CONNECT BY LEVEL	可以使用CTE实现：WITH RECURSIVE
分析函数（窗口函数）	称为分析函数，如 AVG(SALARY) OVER (ORDER BY HIRE_DATE)； AVG(SALARY) OVER (PARTITION BY DEPARTMENT_ID ORDER BY HIRE_DATE)	称为窗口函数，如 AVG(SALARY) OVER (ORDER BY HIRE_DATE)； rank() OVER (PARTITION BY depname ORDER BY salary DESC) 用法与Oracle基本一致
导入导出	exp/imp，datapump	pg_dump
事务超时	不支持	通过参statement_timeout 控制，默认为 0
MERGE	支持	INSERT .. ON CONFLICT
DML…returning	支持	支持 在function返回之前insert或者update的值，可以返回之前各种 DML(insert,delete,update)修改后的值
内部行标识符	ROWID	ctid
CTAS	支持	支持。 CREATE TABLE new_table_name AS query; CREATE TABLE new_table_name (column_name_list) AS query;
闪回查询	支持	支持 需要集成pg_dirtyread插件，默认不开启
SQL Loader	支持	LightDB 21.2提供客户端导入及导出工具ltloader、ltuloader。
外连接	支持 Oracle 专用语法和ANSI 语法 专用语法不支持原生的全外连接	支持。LightDB 21.2支持Oracle外连接语法

if not exists 语 句	不支持	支持，使用IF NOT EXISTS语法进行判断，例如 create table if not exists table_name
INSERT 多表	INSERT ALL INSERT FIRST	使用CTE语句支持该功能
动态 SQL	支持	通过 PREPARE 语句实现
PL/SQL 块中执行 DDL	通过动态 SQL	可直接创建表、临时表等
绑定变量	存储过程/函数自动转换为大写，非动态 SQL 自动优化为绑定变量 可通过参数控制强制启用自动转换为绑定变量	可以自定义变量，plan cache是会话级别的，会话之间不共享plan cache性能更佳
LEVEL 伪列	支持	使用CTE语句支持该功能
ROWNUM伪列	支持	使用窗口函数row_number()实现,如row_number() over() as rownum。后续版本支持原生实现。
CONNECT_BY_***	支持	支持，需要先递归了一次查出结果集，然后再在这个结果集中再递归查询，进行层次标记
定时任务	10g以及之前版本的DBMS_JOB，功能较弱10g以及之后版本的DBMS_SCHEDULER，支持各粒度，链式依赖配置，功能强大、灵活	支持
事务控制	仅支持显示事务控制，不会自动提交	支持隐式事务的概念，可通过参数 autocommit 控制，默认自动提交
语句分隔符	;	通过变量 DELIMITED 控制，尤其在开发 PSM 中使用，如 DELIMITER $$
约束	Primary Key, Foreign Key, Unique, CHECK, NOT NULL约束都支持	Primary Key, Foreign Key, Unique, CHECK, NOT NULL约束都支持
访问其它数据库	dblink 透明网关（需要额外采购）	各种fdw

更详细的语法及编写数据库无关SQL参见《SQL兼容性清单》

6.1 全文检索

LightDB全文检索提供了自然语言搜索的能力。与Oracle不一样，LightDB不需要事先创建词法解析器等数据库对象，而是当作一个普通表和字段来处理。

如下示例，tsvector类型处理过的文本，可搜索，称为‘文档’（Document）；tsquery类型是处理过的查询；@@是匹配操作符。

```
SELECT 'a fat cat sat on a mat and ate a fat rat'::tsvector @@ 'cat &
rat'::tsquery;
 ?column?
----------
 t
```

可以用GIN索引加速全文检索。假设webpage保存了文章的标题（title）和内容（body）：

```
CREATE INDEX pgweb_idx ON pgweb USING GIN (to_tsvector('english', body));
SELECT title
FROM webpage
WHERE to_tsvector(body) @@ to_tsquery('hello');
```

7 应用程序开发

特性	Oracle	LightDB
数据库服务器程序设计语言	PL/SQL	PL/pgSQL
JDBC 支持	支持	支持
ODBC 支持	支持	支持
.NET 支持	支持	支持
存储过程	支持	支持
存储过程的命名参数表示法	支持	支持
游标变量	支持	支持
隐式/显式游标	支持	支持
匿名块	支持	支持
批量收集/绑定	支持	支持
关联数组	支持	支持
嵌套表	支持	支持
VARRAYS 可变数组	支持	支持
级联查询	支持	支持
并行查询	支持	支持
PL/SQL 提供的包	支持	支持
PRAGMA RESTRICT_REFERENCES（程序辅助检验码，检查子程序的纯度（PURITY））	支持	支持
PRAGMA EXCEPTION_INIT (编译异常处理)	支持	支持
PRAGMA AUTONOMOUS_TRANSACTION（自治事务）	支持	支持
用户自定义函数	支持	支持
用户自定义对象	支持	支持
用户自定义异常	支持	支持

7.1 客户端工具

Oracle	LightDB
sql*plus	ltsql
PLSQL Developer、Navicat、Oracle SQL Developer	pgAdmin、Navicat、DBeaver

7.1.1 ltsql

这一部分包含可直接 LightDB 客户端工具或者其他 GUI 工具中执行的常用语句，

不包含只能在存储过程中使用的语句或者表达式。本小节按照使用上的惯例而非字母顺序进行组织。

查看帮助命令

```
DB=# help --总的帮助
DB=# \h --SQL commands级的帮助
DB=# \? --ltsql commands级的帮助
```

按列显示，类似MySQL的\G

```
DB=# \x
Expanded display is on.
```

查看DB安装目录(最好root用户执行)

```
find / -name initdb
```

查看有多少DB实例在运行(最好root用户执行)

```
find / -name postgresql.conf
```

查看DB版本

```
cat $PGDATA/PG_VERSION
ltsql --version
DB=# show server_version;
DB=# select version();
```

查看DB实例运行状态

```
pg_ctl status
```

查看所有数据库

```
ltsql -l
```

查看5432端口下面有多少个

```
ltsql -p 5432 -l
```

查看XX端口下面有多少个DB

```
DB=# \l
DB=# select * from pg_database;
```

创建数据库

```
createdb database_name
DB=# \h create database
```

创建数据库的帮助命令

```
DB=# create database database_name
```

进入某个数据库

```
ltsql –d dbname
DB=# \c dbname
```

查看当前数据库

```
DB=# \c
DB=# select current_database();
```

查看数据库文件目录

```
DB=# show data_directory;
cat $PGDATA/postgresql.conf |grep data_directory
cat /etc/init.d/postgresql|grep PGDATA=
lsof |grep 5432得出第二列的PID号再ps –ef|grep PID
```

查看表空间

```
select * from pg_tablespace;
```

查看语言

```
select * from pg_language;
```

查询所有schema，必须到指定的数据库下执行

```
select * from information_schema.schemata;
SELECT nspname FROM pg_namespace;
\dnS
```

查看表名

```
DB=# \dt
```

只能查看到当前数据库下public的表名

```
DB=# SELECT tablename FROM pg_tables WHERE tablename NOT LIKE 'pg%' AND ta
blename NOT LIKE 'sql_%' ORDER BY tablename;

DB=# SELECT * FROM information_schema.tables WHERE table_name='ff_v3_ff_ba
sic_af';
```

查看表结构

```
DB=# \d tablename

DB=# select * from information_schema.columns where table_schema='public'
and table_name='XX';
```

查看索引

```
DB=# \di

DB=# select * from pg_index;
```

查看视图

```
DB=# \dv
DB=# select * from pg_views where schemaname = 'public';

DB=# select * from information_schema.views where table_schema = 'public';
```

查看触发器

```
DB=# select * from information_schema.triggers;
```

查看序列

```
DB=# select * from information_schema.sequences where sequence_schema = 'p
ublic';
```

查看约束

```
DB=# select * from pg_constraint where contype = 'p'

DB=# select a.relname as table_name,b.conname as constraint_name,b.contype
 as constraint_type from pg_class a,pg_constraint b where a.oid = b.conrel
id and a.relname = 'cc';
```

查看XX数据库的大小

```
SELECT pg_size_pretty(pg_database_size('XX')) As fulldbsize;
```

查看所有数据库的大小

```
select pg_database.datname, pg_size_pretty (pg_database_size(pg_database.d
atname)) AS size from pg_database;
```

查看各数据库数据创建时间：

```
select datname,(pg_stat_file(format('%s/%s/PG_VERSION',case when spcname='
pg_default' then 'base' else 'pg_tblspc/'||t2.oid||'/PG_11_201804061/' end
, t1.oid))).* from pg_database t1,pg_tablespace t2 where t1.dattablespace=
t2.oid;
```

按占空间大小，顺序查看所有表的大小

```
select relname, pg_size_pretty(pg_relation_size(relid)) from pg_stat_user_
```

```
tables where schemaname='public' order by pg_relation_size(relid) desc;
```

按占空间大小，顺序查看索引大小

```
select indexrelname, pg_size_pretty(pg_relation_size(relid)) from pg_stat_
user_indexes where schemaname='public' order by pg_relation_size(relid) de
sc;
```

查看参数文件

```
DB=# show config_file;
DB=# show hba_file;
DB=# show ident_file;
```

查看当前会话的参数值

```
DB=# show all;
```

查看参数值

```
select * from pg_file_settings;
```

查看某个参数值,比如参数work_mem

```
DB=# show work_mem;
```

修改某个参数值,比如参数work_mem

```
DB=# alter system set work_mem='8MB';
```

使用alter system命令将修改postgresql.auto.conf文件，而不是postgresql.conf，这样可以很好的保护postgresql.conf文件，加入你使用很多alter system命令后搞的一团糟，那么你只需要删除postgresql.auto.conf，再执行lt_ctl reload加载postgresql.conf文件即可实现参数的重新加载。

查看是否归档

```
DB=# show archive_mode;
```

查看运行日志的相关配置，运行日志包括Error信息，定位慢查询SQL，数据库的启动关闭信息，checkpoint过于频繁等的告警信息。

show logging_collector;--启动日志收集

show log_directory;--日志输出路径

show log_filename;--日志文件名

show log_truncate_on_rotation;--当生成新的文件时如果文件名已存在，是否覆盖同名旧文件名

show log_statement;--设置日志记录内容

show log_min_duration_statement;--运行XX毫秒的语句会被记录到日志中，-1表示禁用这个功能，0表示记录所有语句，类似mysql的慢查询配置

查看wal日志的配置，wal日志就是redo重做日志

存放在data_directory/pg_wal目录

查看当前用户

```
DB=# \c
DB=# select current_user;
```

查看所有用户

```
DB=# select * from pg_user;
DB=# select * from pg_shadow;
```

查看所有角色

```
DB=# \du
DB=# select * from pg_roles;
```

查询用户XX的权限，必须到指定的数据库下执行

```
select * from information_schema.table_privileges where grantee='XX';
```

创建用户XX，并授予超级管理员权限

```
create user XXX SUPERUSER PASSWORD '123456'
```

创建角色，赋予了login权限，则相当于创建了用户，在pg_user可以看到这个角色

```
create role "user1" superuser; --pg_roles有user1，pg_user和pg_shadow没有user1
alter role "user1" login;--pg_user和pg_shadow也有user1了
```

授权

```
DB=# \h grant
GRANT ALL PRIVILEGES ON schema schemaname TO dbuser;
Grant ALL PRIVILEGES on all tables in schema fds to dbuser;
GRANT ALL ON tablename TO user;
GRANT ALL PRIVILEGES ON DATABASE dbname TO dbuser;
Grant select on all tables in schema public to dbuser;
```

给用户读取public这个schema下的所有表

```
GRANT create ON schema schemaname TO dbuser;
```

给用户授予在schema上的create权限，比如create table、create view等

```
GRANT USAGE ON schema schemaname TO dbuser;
Grant select on schema public to dbuser;
```

报错ERROR: invalid privilege type SELECT for schema

--USAGE：对于程序语言来说，允许使用指定的程序语言创建函数;对于Schema来说，允许查找该Schema下的对象;对于序列来说，允许使用currval和nextval函数;对于外部封装器来说，允许使用外部封装器来创建外部服务器;对于外部服务器来说，允许创建外部表。

查看表上存在哪些索引以及大小

```
select relname,n.amname as index_type from pg_class m,pg_am n where m.rela
m = n.oid and m.oid in(select b.indexrelid from pg_class a,pg_index b wher
e a.oid = b.indrelid and a.relname = 'cc');
select c.relname,c2.relname, c2.relpages*8 as size_kb FROM pg_class c, pg_
class c2, pg_index where c.relname ='cc' AND c.oid =i.indrelid AND c2.oid
=i.indexrelid ORDER BY c2.relname;
```

查看索引定义

```
select b.indexrelid from pg_class a,pg_index b where a.oid = b.indrelid an
d a.relname = 'cc';
select pg_get_indexdef(b.indexrelid);
```

查看过程函数定义

```
select oid,* from pg_proc where proname = 'insert_platform_action_exist';
--oid = 24610
select * from pg_get_functiondef(24610);
```

查看表大小(不含索引等信息)

```
select pg_relation_size('cc'); --368640 byte
select pg_size_pretty(pg_relation_size('cc')) --360 kB
```

查看表所对应的数据文件路径与大小

```
SELECT pg_relation_filepath(oid), relpages FROM pg_class WHERE relname = '
empsalary';
```

```
posegresql查询当前lsn
```

用到哪些方法：

```
apple=# select proname from pg_proc where proname like 'pg_%_lsn';
proname
---------------------------------
pg_current_wal_flush_lsn
pg_current_wal_insert_lsn
pg_current_wal_lsn
pg_last_wal_receive_lsn
pg_last_wal_replay_lsn
```

查询当前的lsn值：

```
apple=# select pg_current_wal_lsn();
pg_current_wal_lsn;
0/45000098
```

查询当前lsn对应的日志文件

```
select pg_walfile_name('0/1732DE8');
```

7.2 存储过程/函数

注：非极端响应时间应用，不推荐使用存储过程和函数。要使用的话，优先使用匿名块嵌入mybatis。

Oracle定义存储过程的关键字为procedure，基本语法如下：

```
create or replace procedure 存储过程名(参数1 类型，参数2 out 类型……)
    as
        变量名     类型;
    begin
        程序代码体
    end;
```

举例如下：

（无参数）

```

create or replace procedure p1
--or replace代表创建该存储过程时，若存储名存在，则替换原存储过程，重新创建
--无参数列表时，不需要写()
as
```

```
begin
    dbms_output.put_line('hello world');
end;
```

有参有返

```
create or replace procedure p2 (name in varchar2,age int,msg out varchar2)
--参数列表中，声明变量类型时切记不能写大小，只写类型名即可，例如参数列表中的name变量的声明
--参数列表中，输入参数用in表示，输出参数用out表示，不写时默认为输入参数。
--输入参数不能携带值出去，输出参数不能携带值进来，当既想携带值进来，又想携带值出去，可以用in out
as
begin
    msg:='姓名'||name||',年龄'||age;
end;
```

LightDB则将函数和存储过程合为一体，不再明确区分存储过程与函数。定义函数（存储过程 ）的关键字为function，基本语法如下：

```
create or replace function 过程名(参数名 参数类型,…..)
returns 返回值类型
as
  $body$
Declare
       变量名变量类型;
Begin
        return 变量名;  //存储过程中的返回语句
End;
 $body$
 Language plpgsql;
```

举例如下：

```
CREATE OR REPLACE FUNCTION public.udf_station_in_step5_new()
  RETURNS void AS
$BODY$
DECLARE
    i integer := 0;
    tmp_reportdate timestamp ARRAY[100];
```

```
    res_slno integer ARRAY[100];
    res_reportdate timestamp ARRAY[100];
    v_cur_slno integer;
    v_cur_reportdate timestamp;
    v_sql varchar(500);
    cur_list cursor for select slno, reportdate from
ods_t_station_in_single order by reportdate;
begin
    open cur_list;
    fetch cur_list into v_cur_slno, v_cur_reportdate;
    while found loop
        i := i+1;
        tmp_slno := v_cur_slno;
        tmp_reportdate[i] := v_cur_reportdate;
        fetch cur_list into v_cur_slno, v_cur_reportdate;
        while found loop
            if v_cur_slno > tmp_slno and v_cur_slno < tmp_slno+10 then
                i := i+1;
                tmp_slno := v_cur_slno;
                tmp_reportdate[i] := v_cur_reportdate;
                fetch cur_list into v_cur_slno, v_cur_reportdate;
            else
                if res_reportdate is null or
array_length(tmp_reportdate,1) > array_length(res_reportdate,1) then
                    res_reportdate := tmp_reportdate;
                end if;
                i := 0;
                tmp_reportdate := null;
                exit;
            end if;
        end loop;
    end loop;
    close cur_list;
  --根据最长的序列挑选出最可能正确的数据
  execute  'delete from ods_t_station_in_single2';
  if(res_reportdate is not null) then
      for i in 1 .. array_length(res_reportdate,1) loop
        v_sql := 'insert into ods_t_station_in_single2 select * from
ods_t_station_in_single where reportdate ='''||res_reportdate[i]||'''';
        execute v_sql;
```

```
      end loop;
  end if;
  execute 'delete from ods_t_station_in_single';
  insert into ods_t_station_in_single select * from
ods_t_station_in_single2 order by reportdate;
  execute 'delete from ods_t_station_in_single2';
  exception
  when QUERY_CANCELED then
           raise 'udf_station_in_step5(0)';
  commit;
end;
$BODY$
  LANGUAGE plpgsql;
```

oracle 的函数

```
CREATE OR REPLACE FUNCTION cs_fmt_browser_version(v_name varchar2,
v_version varchar2)
RETURN varchar2 IS BEGIN
IF v_version IS NULL THEN RETURN v_name;
END IF;
RETURN v_name || '/' || v_version; END;
```

LightDB 函数

```
CREATE OR REPLACE FUNCTION cs_fmt_browser_version(v_name varchar,
v_version varchar)
RETURNS varchar AS $$ BEGIN
IF v_version IS NULL THEN RETURN v_name;
END IF;
RETURN v_name || '/' || v_version; END;
$$ LANGUAGE plpgsql;
```

7.2.1 匿名块

匿名块的优点在于它能够很方便地执行判断，而且无需先创建存储过程或函数。对于确保DDL可重复执行来说具有非常大的优势，对于产品系统来说是非常有价值的。

```
LightDB=> DO $$
DECLARE
```

```
  v_sal_chk DOUBLE PRECISION;
  v_emp_work_years DOUBLE PRECISION;
  v_sql_cmd CHARACTER VARYING(2000);
  v RECORD;
BEGIN
  FOR v IN
    SELECT employee_id, CONCAT_WS('', first_name, ' ', last_name) AS
      emp_name, hire_date, salary
      FROM employees
  LOOP
    v_emp_work_years := EXTRACT (YEAR FROM now()) - EXTRACT (YEAR FROM
      v.hire_date);
    IF v_emp_work_years >= 10 AND v.salary <= 6000 THEN
      RAISE DEBUG USING MESSAGE := CONCAT_WS('', 'Consider a Salary
        Raise for: ', v.emp_name);
    END IF;
  END LOOP;
  EXCEPTION
   WHEN others THEN
   RAISE DEBUG USING MESSAGE := CONCAT_WS('', 'CODE ERR: ',
     SQLERRM);
END $$;
```

7.3 JDBC驱动包

特性	oracle	LightDB
JDBC 驱动包	ojdbc8.jar	postgresql-42.2.12.jar
maven 仓库	<dependency> <groupId>com.oracle.database.jdbc</groupId> <artifactId>ojdbc8-production</artifactId> <version>21.1.0.0</version> <type>pom</type> </dependency>	<dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <version>42.2.12</version> </dependency> 注：不要使用 jre6 和 jre7 版本。
JDBC 驱动类	oracle.jdbc.OracleDriver	org.postgresql.Driver
JDBC URL 格式	jdbc:oracle:thin:@//host:port/ServiceName jdbc:oracle:thin:@host:port:SID jdbc:oracle:thin:@TNSName	jdbc:postgresql:database jdbc:postgresql: jdbc:postgresql://host/database jdbc:postgresql://host/ jdbc:postgresql://host:port/database jdbc:postgresql://host:port/
简单示例	Class.forName("oracle.jdbc.OracleDriver");	Class.forName("org.postgresql.Driver");

	Connection connection=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:test", "用户名", "密码");	Connection connection = DriverManager.getConnection(jdbc:postgresql://localhost:5432/test"用户名", "密码");

7.3.1 MyBatis示例

插入自增返回值: https://blog.csdn.net/llmys/article/details/80745441

```
<insert id="insertUser">
    <selectKey resultType="int"  order="AFTER" keyProperty="pid" >
        SELECT currval('tbl_user_pid_seq'::regclass) AS pid
    </selectKey>
    insert into tbl_user(name, age) values(#{name}, #{age})
</insert>
```

7.3.2 调用匿名块

在java中调用多语句匿名块可以极大的提高性能，减少网络交互、数据来回传输，且相比存储过程来说，侵入性大大降低。

LightDB支持使用jdbc调用匿名块，如下示例：

```
//获取数据库连接
conn = getConnection();
//创建statement
statement = conn.createStatement();
//1.调用DO匿名结构块
statement.execute("DO\n" +
"$$DECLARE i record;\n" +
"BEGIN\n" +
"DELETE FROM cust WHERE id = 5;\n" +
"END$$;");
//2.调用命名结构块
statement.execute("select query()");
```

7.3.3 JdbcTemplate示例

增删改查示例代码 https://www.cnblogs.com/hellowhy/p/7767968.html

插入自增返回值:

```
JdbcTemplate jdbcTemplate = new JdbcTemplate(datasource);
KeyHolder keyHolder = new GeneratedKeyHolder();
int temp = jdbcTemplate.update("insert into table values(id,name)");
if(temp>0){
Long id = keyHolder.getKey().longValue();
}
```

删除：

```
JdbcTemplate jdbcTemplate = new JdbcTemplate(datasource);
jdbcTemplate.update("delete from table where id = $1");
```

修改：

```
JdbcTemplate jdbcTemplate = new JdbcTemplate(datasource);
jdbcTemplate.update("update table set id = $1 where id = $2");
```

查询单个：

```
JdbcTemplate jdbcTemplate = new JdbcTemplate(datasource);
Clazz<T> resultObject =
jdbcTemplate. queryForObject("select * from table where id = $1",Clazz<T>);
```

查询列表:

```
JdbcTemplate jdbcTemplate = new JdbcTemplate(datasource);
List<Clazz<T>> resultList =
jdbcTemplate. queryForList("select * from table",Clazz<T>);7.4 C客户端库
```

7.4 C客户端库

libpq是LightDB的C语言编程库，它提供了一套接口用于连接与查询。同时lipq也是很多其它语言接口的底层库。

连接

```
PGconn     *conn;
conn = PQconnectdb(conninfo);
```

设置选项：

```
res = PQexec(conn, "SET search_path = testlibpq3");
```

增删改查

```
paramValues[0] = "LightDB";


res = PQexecParams(conn,
                   "SELECT * FROM test1 WHERE t = $1",
                   1,       /* 1个参数 */
                   NULL,
                   paramValues,
                   NULL,
                   NULL,
                   1);
```

事务

```
res = PQexec(conn, "BEGIN"); /* 开始事务 */
res = PQexec(conn, "DECLARE myportal CURSOR FOR select * from
pg_database");
res = PQexec(conn, "FETCH ALL in myportal");
/* ……处理结果 */
```

```
res = PQexec(conn, "CLOSE myportal");
res = PQexec(conn, "END"); /* 结束事务 */
```

7.5 其它编程语言

除了libpq，其它语言如C#、C++、Delphi、Perl、PHP、Python、go等。

7.6 大数据导入导出

有多种方式可以把数据导入LightDB。

1. ltsql中使用\COPY命令数据导入导出
2. 如果数据使用lt_dump导出，则可以使用lt_restore导入
3. Java程序可以使用JDBC中的CopyManager类实现数据导入导出

8 数据库管理

数据库管理	Oracle	LightDB
管理客户端	SQL*Plus	ltsql
批量数据加载器	SQL*Loader	ltloader
批量数据导出	无	ltunloader
企业管理	Oracle Enterprise Manager	LightDB Enterprise Manager
系统目录视图	支持	支持
时间点恢复 (PITR)	支持	支持
在线备份	支持	支持

9 数据库监控

LightDB数据库监控系统，是对数据库各种运行指标进行全方位实时监控的产品。系统能够发现和识别数据库异常以及潜在的性能问题，并及时将数据库异常报告给管理员，通过针对各项运行指标的统计分析报表，帮助管理员、运维人员、决策者多视角了解数据库的运行状态，从而更好的应对数据库的需求及规划。

当前版本的LightDB，提供了下列数据库监控功能：

1. pgcenter：类似于top的LightDB实时监控命令行。
2. PWR：类似于Oracle awr的LightDB实例分析报告。

注：21.2将支持LightDB-EM数据库企业管理平台。

10 安全性

安全特性	Oracle	LightDB
身份系统支持	支持 LDAP, SSL, RADIUS, PAM, Kerberos, GSSAPI, SSPI	支持 LDAP, SSL, RADIUS, PAM, Kerberos, GSSAPI, SSPI
数据库连接加密	支持	支持
数据库连接白名单	支持 –使用连接前触发器	支持
数据库连接黑名单	支持 –使用连接后触发器	支持
密码配置文件	支持	支持
服务器代码混淆	支持	支持
ANSI 标准 SQL 授权/撤销	支持	支持
列级权限	支持	支持
用户/组/角色支持	支持	支持
虚拟专用数据库（VPD）	支持	支持
视图安全屏障	不支持	支持
数据加密工具包	支持	支持
透明加密(TDE,FDE)	支持	LightDB 支持

10.1 数据加密选项

以下数据加密选项根据应用的需要提供的保护级别提供了不同的级别和粒度：

pgcrypto

- Postgres crypto模块
- 应用于选定的表列
- 无法搜索或索引加密字段
- 必须在创建表时应用加密，因此需要提前规划
- 应用程序必须处理加密/解密以便与数据库交换保持加密
- DBA 无法清楚地看到数据

TED/FED加密（LightDB提供的文件级别加密方案）

透明数据加密（通常缩写为 TDE）是 Microsoft、IBM 和 Oracle 用来加密数据库文件的一种技术。 TDE 提供文件级别的加密。 TDE解决了保护静态数据的问题，对硬盘上的数据库以及备份介质上的数据库进行加密。

10.2 审计

为配合政府、金融部门的审计认证，LightDB提供了强大的审计工具。pgaudit插件通过LightDB的日志接口，提供了会话和对象级别的审计日志。

相关的设置只能由数据库超级账户来操作。如果普通账户可以修改设置，那么审计就没有意义了。

会话审计

例1：记录DML和DDL操作日志；及所有关系（表）的DML操作日志

```
set pgaudit.log = 'write, ddl';
set pgaudit.log_relation = on;
```

例2：记录所有日志，但是不包括 DISCARD, FETCH, CHECKPOINT, VACUUM, SET；审计日志级别设为notice

```
set pgaudit.log = 'all, -misc';
set pgaudit.log_level = notice;
```

数据库对象审计

对象审计只支持SELECT, INSERT, UPDATE 和 DELETE，相比pgaudit.log = 'read,write'，它提供了更新的控制粒度。下面示例设置了角色auditor的审计范围：在public.account关系上的select和delete操作。

```
set pgaudit.role = 'auditor';

grant select, delete
   on public.account
   to auditor;
```

11 LightDB vs. Oracle 高可用

11.1 LightDB分布式集群及高可用方案

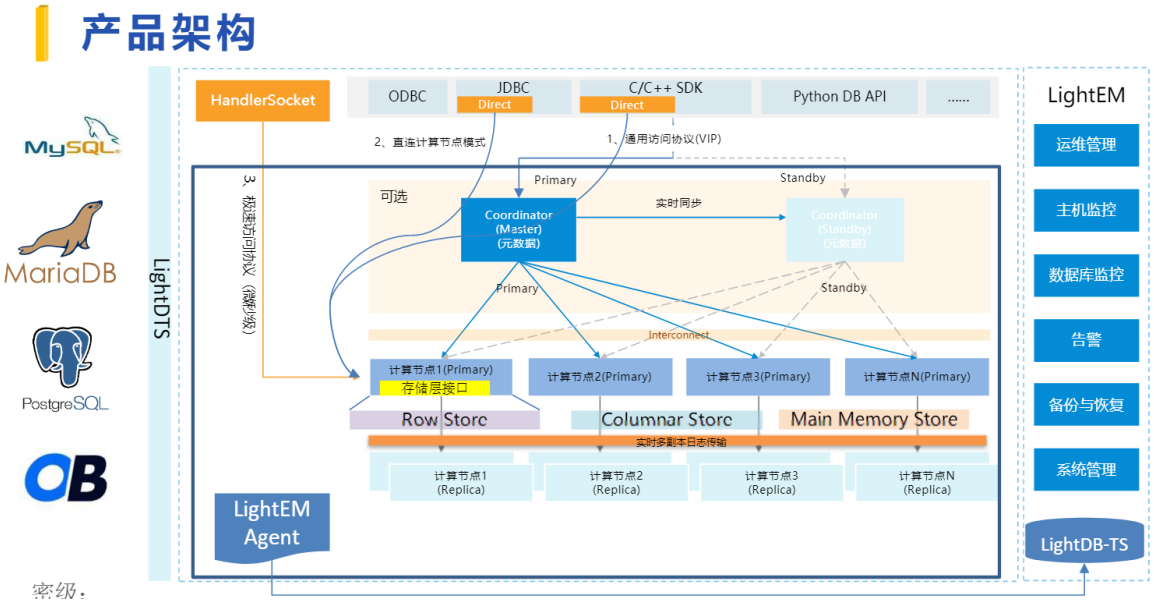

图 LightDB架构图

11.2 功能对比

高可用特性	Oracle	LightDB
Data Guard 双机热备	支持	支持 使用 hot_standby 异步模式可以实现
Active Data Guard	支持	支持 使用 hot_standby 同步模式可以实现，LightDB 默认包含 HA 实现。
Flashback Query 闪回查询	支持	不支持
Flashback Table, Database and Transaction Query 闪回表、数据库，事务查询	支持	不支持
Backup and Recovery Tools 备份恢复工具	支持	支持
Point in Time Recovery 时间点恢复	支持	支持

可扩展性	Oracle	LightDB
连接池	支持。但是很少有人使用。	支持。通过 pgBouncer，。LightDB 包含连接池选件。
分布式集群	支持 Oracle Real Application Clusters (RAC)，一种 shared-everything 的架构用	支持 采用 shared-nothing 的高可用架构，支持分布式事务。

	于性能、高可用性和读取扩展.	
内存数据库	支持	支持。LightDB 22c 支持。
多主机读写解决方案	Oracle GoldenGate	双向逻辑复制
列存	不支持	支持 外部数据源 cstore_fdw， oss_fdw 的列式存储
CPU 和 I/O 资源限制	支持	支持

12 Oracle到LightDB迁移工具

12.1 Ora2Pg 特性介绍

它不仅能够迁移表结构，还包括存储过程、视图、函数，能够极大的降低迁移成本。

Ora2Pg 特性包括：

1. 导出完整的schema，包括表、视图、序列、索引、主键、外键、及相关约束
2. 导出用户和组的权限
3. 导出range/list分区及子分区
4. 导出函数、触发器、存储过程
5. 导出完整的数据，或根据where条件导出数据
6. 把Oracle Blob转换为PG BYTEA
7. 导出Oracle视图，作为PG表
8. 导出Oracle自定义数据类型

12.2 Ora2Pg 使用方法

1、下载

地址：https://github.com/darold/ora2pg/releases

2、条件检查

必要条件

① 需要安装Oracle客户端和Oracle相关工具

```
rpm -ivh oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64.rpm
rpm -ivh oracle-instantclient12.2-devel-12.2.0.1.0-1.x86_64.rpm
rpm -ivh oracle-instantclient12.2-jdbc-12.2.0.1.0-1.x86_64.rpm
rpm -ivh oracle-instantclient12.2-sqlplus-12.2.0.1.0-1.x86_64.rpm
```

② Perl distribution (perl 5.10或更高版本)

③ DBI Perl module > 1.614

④

DBD::Oracle Perl module（迁移Oracle,需要安装Oracle客户端并配置环境变量）

DBD::MySQL Perl module（迁移mysql，需要安装MySQL客户端)

⑤某些PERL发行版，可能需要安装Time::HiRes Perl模块

可选条件

默认Ora2Pg dump导出到文本文件，使用ltsql应用到LightDB数据库，如果你想在线应用，需要安装DBD::Pg 模块

- Ora2Pg可以压缩导出文件（gzip or bzip2),需要安装Compress::Zlib Perl 模块，程序bzip2需要添加到PATH变量

3、安装Ora2Pg （linux环境下）

```
tar xjf ora2pg-x.x.tar.bz2
cd ora2pg-x.x/
perl Makefile.PL
make && make install
```

默认安装位置/etc/ora2pg/

4、安装DBD::Oracle

配置环境变量

```
[root@emr ~]#vi ~/.bash_profile
export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=$ORACLE_BASE/product/12.2.0/dbhome_1
export PATH=$ORACLE_HOME/bin:$PATH
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/lib:/usr/lib

[root@emr ~]#source   ~/.bash_profile
```

通过CPAN安装

```
#perl -MCPAN -e shell
cpan> get DBD::Oracle
cpan> quit
cd ~/.cpan/build/DBD-Oracle*
export LD_LIBRARY_PATH=/u01/app/oracle/product/12.2.0/dbhome_1/lib
export ORACLE_HOME=/u01/app/oracle/product/12.2.0/dbhome_1perl Makefile.PL
make
make install
```

也可手动下载安装DBD::Oracle。下载路径：https://metacpan.org/release/DBD-Oracle。

预先安装依赖：

```
yum install perl-DBI perl-DBD-Pg perl-ExtUtils-MakeMaker gcc
```

5、配置文件配置

① Oracle连接

```
ORACLE_HOME     /u01/app/oracle/product/12.2.0/dbhome_1
ORACLE_DSN
dbi:Oracle:host=oradb_host.myhost.com;sid=DB_SID;port=1521
or
dbi:Oracle:DB_SID
```

比如

```
ORACLE_DSN      dbi:Oracle:host=172.16.1.151;sid=ipdb;port=1521
对于mysql,DSN如下
        dbi:mysql:host=192.168.1.10;database=sakila;port=3306
ORACLE_USER     system
ORACLE_PWD      #######
```

使用Oracle服务对数据加密

```
# Configure encryption of connections to Oracle
SQLNET.ENCRYPTION_CLIENT = required
SQLNET.ENCRYPTION_TYPES_CLIENT = (AES256, RC4_256)
SQLNET.CRYPTO_SEED = 'should be 10-70 random characters'
```

测试连接

```
ora2pg -t SHOW_VERSION -c config/ora2pg.conf
```

② Oracle导出schema

```
SCHEMA  APPS
pg_schema APPS
type    table
```

```
JOBS     5  ##导出并行度，支持copy、function、procedure
PARALLEL_TABLES  2  ###并行处理的表的个数，实际的进程数为PARALLEL_TABLES*JOBS，根据实际服务器cpu内核设置
```

③ 限制导出对象

```
### 导出EMPLOYEES 、COUNTRIES 和以SALE开头、GEOM_SEQ结尾的表；可使用正则匹配
ALLOW  EMPLOYEES SALE_.* COUNTRIES .*_GEOM_SEQ
```

```
### 排除某些对象
EXCLUDE  EMPLOYEES TMP_.* COUNTRIES
```

```
### 可使用where条件过滤数据
WHERE  ROWNUM < 1000
```

```
### 开启此参数可使遇到错误时继续执行导入
LOG_ON_ERROR
```

④ postgresql导入

ltsql mydb < output.sql

DATA_LIMIT

批量处理的数据条目数，默认10000，可视情况增加此参数以提高性能

OUTPUT

导出的文件名

OUTP_DIR

导出的文件路径

STOP_ON_ERROR

设置为0，可以遇到错误时不异常退出

PG_DSN

```
dbi:Pg:dbname=pgdb;host=localhost;port=5432
```

比如

```
PG_DSN          dbi:Pg:dbname=jhhis;host=172.16.0.20;port=5432
```

```
PG_USER         jhhis
PG_PWD          ####
```

⑤导出控制

```
SKIP
- fkeys: turn off foreign key constraints
- pkeys: turn off primary keys
- ukeys: turn off unique column constraints
- indexes: turn off all other index types
- checks: turn off check constraints
```

6、使用

```
####导出表定义
ora2pg -c ora2pg.conf -t TABLE -n ORATEXT -N ORATEXT  -o TB_ORATEXT.sql

#####COPY表数据
ora2pg -c ora2pg.conf -t COPY   -j  6   -n ORATEXT  -N ORATEXT
```