MySQL开发与优化

杨涛-资深顾问
上海爱可生信息技术有限公司
MSN： yueliangdao0608@gmail.com
EMAIL: david.yang@actionsky.com

MySQL开发与优化

- 表设计
- 索引规划
- 语句优化
- 存储过程
- 触发器
- 视图

表设计-目录

- 命名规则
- 字段类型
- 引擎选择
- 编码选择

表设计-命名规则

- 保留词
- 多字节字符

表设计-字段类型

- 数值类型
- 字符类型
- 二进制类型
- 时间类型
- 其他类型

数值类型–整型

名称	大小	有符号	无符号
TINYINT	1 字节	-128 to 127	0 to 255
SMALLINT	2 字节	-32,768 to 32,767	0 to 65,535
MEDIUMINT	3 字节	-8,388,608 to 8,388,607	0 to 16,777,215
INT	4 字节	-2,147,483,648 to 2,147,483,647	0 to 4,294,967,295
BIGTIN	8 字节	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0 to 18,446,744,073,709,551,615

数值类型–非精确浮点数

名称	大小	范围
FLOAT	4 字节	-3.402823466E+38 to -1.175494351E-38, 0 and 1.175494351E-38 to 3.402823466E+38
DOUBLE REAL DOUBLE PRECISION	8 字节	-1.7976931348623157E+308 to -2.2250738585072014E-308, 0 and 2.2250738585072014E-308 to 1.7976931348623157E+308

数值类型–精确浮点数

- DECIMAL
- NUMERIC
- DEC

字符类型

名称	大小	范围
CHAR(M)	M+1 字节	255个字符
VARCHAR(M)	M+1 OR M+2 字节	65,535 字节
TINYTEXT	真实字符+1个字节	255 字节
TEXT	真实字符+2个字节	65,535 字节
MEDIUMTEXT	真实字符+3个字节	16,777,215 字节
LONGTEXT	真实字符+4个字节	4,294,967,295 字节

二进制类型

名称	大小	范围
BINARY(M)	M字节	255个字符
VARBINARY(M)	M+1 OR M+2 字节	65,533 字节
TINYBLOB	真实字节+1个字节	255 字节
BLOB	真实字节+2个字节	65,535 字节
MEDIUMBLOB	真实字节+3个字节	16,777,215 字节
LONGBLOB	真实字节+4个字节	4,294,967,295 字节

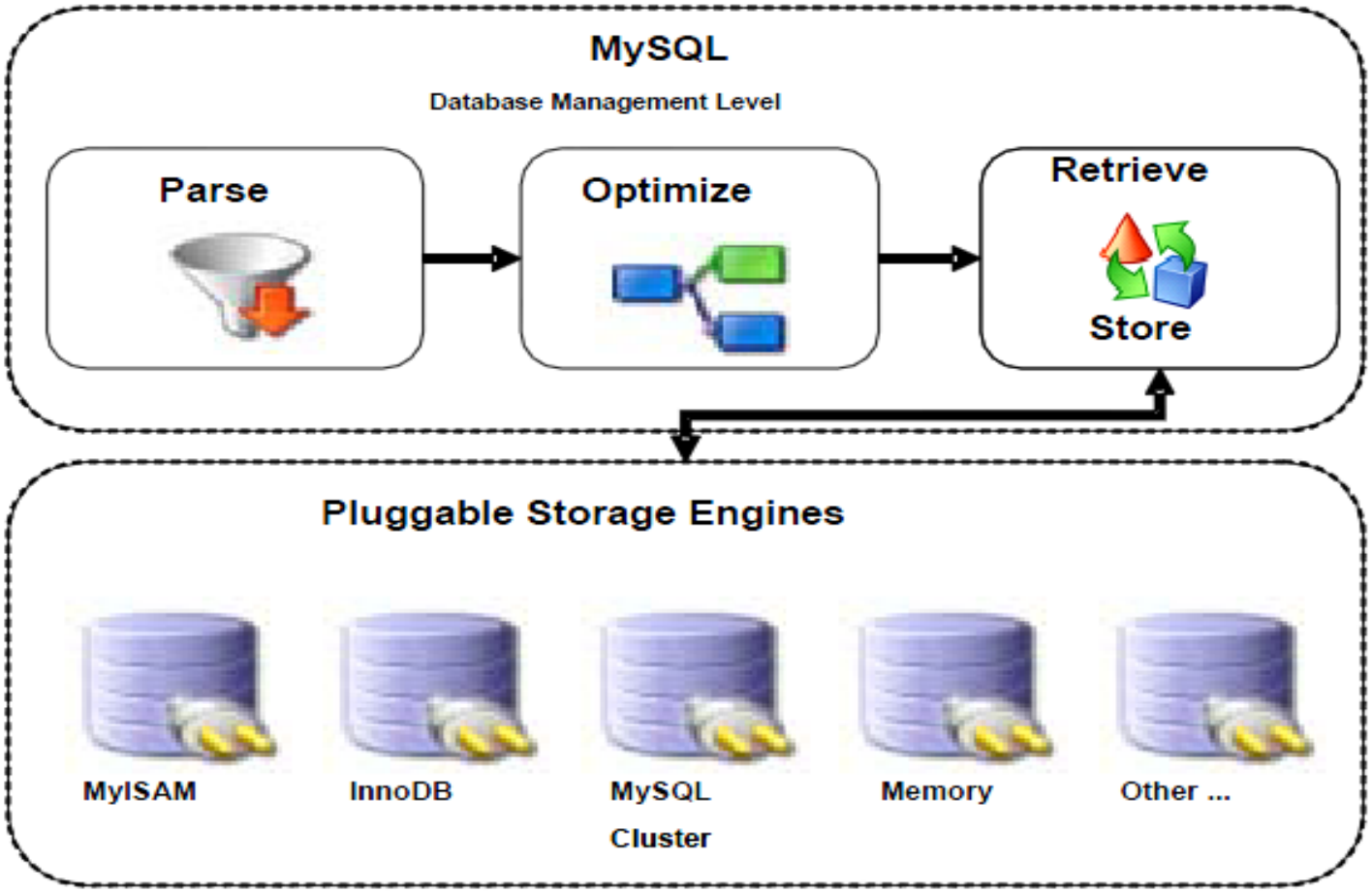
时间类型

名称	大小	范围
DATE	3 字节	'1000-01-01' to '9999-12-31'
TIME	3 字节	'-838:59:59' to '838:59:59‘
DATETIME	8 字节	'1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP	4 字节	'1970-01-01 00:00:00' to mid-year 2037
YEAR	1 字节	1901 to 2155 (for YEAR(**4**)), 1970 to 2069 (for YEAR(**2**))

其他类型–BIT

- 比特位？
- BIT(1-64)
- 大小(最大9个字节)

引擎选择


MySQL
Database Management Level
Parse
Optimize
Retrieve
Store
Pluggable Storage Engines
MyISAM
InnoDB
MySQL
Cluster
Memory
Other ...

常用引擎

- MyISAM
  快读
  高压缩比
  表锁
- InnoDB
  事务支持
  外键支持
  行锁
  热备支持
- Memory
  内存数据
  表锁

常用引擎总结

	MyISAM	Memory	InnoDB
用途	快读	内存数据	完整的事务支持
锁	全表锁定	全表锁定	多种隔离级别的行所
持久性	基于表恢复	无磁盘I/O，无可持久性	基于日志的恢复
事务特性	不支持	不支持	支持

最可靠性引擎

- NDB (MySQL Cluster)
  - 支持事务
  - 高并发写
  - KEY→VALUE 可持续化
  - 99.999% 的高靠性

其他

- ARCHIVE

  比MYISAM更高的压缩比

- CSV

  EXCEL 和 MySQL 之间沟通的桥梁

- BLACKHOLE

  缓解REPLICATION中主的压力

编码选择

- 单字节?
- 多字节?

索引规划

- 唯一索引
- 普通索引
- 部分索引
- 聚簇索引
- 外键

例子

```
CREATE TABLE `CountryLanguage` (
`CountryCode` char(3) NOT NULL default '',
`Language` char(30) NOT NULL default '',
`IsOfficial` enum('T','F') NOT NULL default 'F',
`Percentage` float(4,1) NOT NULL default '0.0',
PRIMARY KEY (`CountryCode`,`Language`),
Key idx_language(`Language`),
Key idx_language_patial (`Language(10)`)
) ENGINE=MyISAM COMMENT='List Languages Spoken'
```

	MyISAM	MEMEORY	InnoDB
支持索引类型	B-tree/Full Text/R-tree	Hash/B-tree	B-tree

语句优化

- 响应时间？
  - 执行时间
  - 传送时间

读语句优化

- 一些好的习惯1

```
SELECT * FROM t WHERE YEAR(d) >= 1994;
```

→

```
SELECT * FROM t WHERE d >= '1994-01-01';
```

```
SELECT * FROM Country ,CountryLanguage
WHERE Country.Code = CountryLanguage.CountryCode;
```

→

```
SELECT * FROM Country JOIN CountryLanguage
ON Country.Code = CountryLanguage.CountryCode;
```

```
SELECT * FROM t WHERE id = '19';
```

→

```
SELECT * FROM t WHERE id = 19;
```

```
SELECT * FROM t WHERE length(column_t) = 5;
```

→

```
SELECT * FROM t WHERE column_length=5;
```

读语句优化

- 一些好的习惯2

SELECT * FROM t WHERE name **LIKE '%de%'**

→

SELECT * FROM t WHERE name **LIKE 'de%'**

SELECT * FROM t WHERE **name >= 'de' AND name < 'df'**

SELECT * FROM t WHERE 1;

→

SELECT * FROM t WHERE 1 **LIMIT 10;**

SELECT * FROM Country WHERE Name **LIKE 'M%';**

→

SELECT **Name** FROM Country WHERE Name **LIKE 'M%';**

示例7

- 修改语句后

```
SELECT DISTINCT a.stop_time,
CASE WHEN b.ne_id = '1403000000011001' THEN b.tank END AS '1403000000011001',
...
CASE WHEN b.ne_id = '1403000000011006' THEN b.tank END AS '1403000000011006'
FROM tb1 a INNER JOIN tb1 b USING (stop_time)
WHERE a.stop_time>'2010-08-28 00:30:00' AND a.stop_time<'2010-08-28 01:45:00'
ORDER BY a.stop_time ASC;
```

对b表扫描次数减小到1!

摘要表

- 优点：

  减少对原磁盘表的访问。

- 缺点：

  要做适当的更新。

写语句优化

- INSERT

```
INSERT INTO t (id, name) VALUES(1,'Bea');
INSERT INTO t (id, name) VALUES(2,'Belle');
INSERT INTO t (id, name) VALUES(3,'Bernice');
```

→

```
INSERT INTO t (id, name) VALUES(1,'Bea'), (2,'Belle'),(3,'Bernice');
```

→

```
START TRANSACTION;
  INSERT INTO t (id, name) VALUES(1,'Bea');
  INSERT INTO t (id, name) VALUES(2,'Belle');
  INSERT INTO t (id, name) VALUES(3,'Bernice');
COMMIT;
```

INSERT

- 两者的性能对比

	第一种	第二种
Real	12m17.151s	0m33.368s
User	0m 1.412s	0m 0.744s
Sys	0m 34.080s	0m 0.419s

UPDATE

- 注意索引：

```
UPDATE update_ext
SET create_time= now()
WHERE create_time= '2009-07-05 08:46:48';
```

- 注意优化以下语句：

```
SELECT * FROM update_ext
WHERE create_time= '2009-07-05 08:46:48';
```

DELETE

- SQL

  DELETE FROM table_name;

  TRUNCATE TABLE table_name;

  DROP TABLE table_name;

- 删除物理文件

存储过程

- 存储函数
- 动态SQL
- 异常处理

存储函数-例子1

```
DELIMITER $$
CREATE FUNCTION `t_girl`.`func_rand_string`(f_num TINYINT UNSIGNED, f_type TINYINT UNSIGNED)
RETURNS VARCHAR(32)
BEGIN
        -- Translate the number to letter.
        -- 1 代表纯字符。
        -- 2 代表纯数字。
        -- 3 代表数字与字符

       DECLARE i INT UNSIGNED DEFAULT 0;
       DECLARE v_result VARCHAR(255) DEFAULT ‘’;
       WHILE i < f_num DO
                IF f_type = 1 THEN
                                SET v_result = concat(v_result,char(97+ceil(rand()*25)));
                ELSEIF f_type=2 THEN
                                SET v_result = concat(v_result,char(48+ceil(rand()*9)));
               ELSEIF f_type=3 THEN
                                SET v_result = concat(v_result,substring(replace(uuid(),'-',''),i+1,1));
               END IF;
               SET i = i + 1;
        END WHILE;
        RETURN v_result;
END$$
DELIMITER ;
```

存储函数-例子2

```
DELIMITER $$
CREATE    FUNCTION `test`.`t-cursor`()    RETURNS VARCHAR(255)
 BEGIN
    DECLARE v_result VARCHAR(255) DEFAULT '';
    DECLARE v_id INT DEFAULT 0;
    DECLARE done INT DEFAULT 0;
    DECLARE c CURSOR FOR SELECT id FROM event WHERE 1 LIMIT 5;
    DECLARE CONTINUE HANDLER FOR NOT FOUND
    BEGIN
      SET done = 1;
    END;
    OPEN c;
    loop1:LOOP
      IF done = 1 THEN
        LEAVE loop1;
      END IF;
      FETCH c INTO v_id;
      SET v_result = CONCAT(v_result,',',v_id);
      END LOOP loop1;
    CLOSE c;
    SET v_result = SUBSTR(v_result,2,LENGTH(v_result));
    RETURN v_result;
END$$
DELIMITER ;
```

动 态SQL

- 两种方式
  - -- 占位符 ?
  - -- 直接赋值。

动态SQL-例子1

```
DROP PROCEDURE IF EXISTS d_ytt2;
DELIMITER //
CREATE PROCEDURE d_ytt2(
        field1 VARCHAR(20),
        table1 VARCHAR(20),
        val1 INT,
        val2 INT,
        val3 VARCHAR(100)
)
BEGIN
        DECLARE var INTEGER DEFAULT 1;
        SET @a=val1, @b=val2, @c=val3, @d=field1;
        SET @stmt_text=concat("select ", field1, "  from ", table1, " WHERE ", field1 ," = ?" );
         PREPARE stmt FROM @stmt_text;
        EXECUTE stmt USING @c;
        DROP PREPARE stmt;
END;
//
DELIMITER ;
```

动态SQL-例子2

```
DROP PROCEDURE IF EXISTS d_ytt3;
DELIMITER //
CREATE PROCEDURE d_ytt3(
    field1 VARCHAR(20),
    table1 VARCHAR(20),
    val1 INT,
    val2 INT,
    val3 VARCHAR(100)
)
BEGIN
    DECLARE var INTEGER DEFAULT 1;
    SET @a=val1, @b=val2, @c=val3, @d=field1;
    SET @stmt_text=concat("select ", field1, "  from ", table1, " WHERE ", field1 ,"=", val3);
    PREPARE stmt FROM @stmt_text;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END;
//
DELIMITER ;
```

异常处理

- DECLARE *handler_type* HANDLER FOR *condition_value*[,...] *statement*

  handler_type:

  CONTINUE
  | EXIT
  | UNDO

  condition_value:

  SQLSTATE [VALUE] *sqlstate_value*
  | *condition_name*
  | SQLWARNING
  | NOT FOUND
  | SQLEXCEPTION

异常处理-例子

```
DELIMITER //
CREATE PROCEDURE ytt_AddData(IN v_id BIGINT UNSIGNED)
BEGIN
	DECLARE uid BIGINT UNSIGNED;
	DECLARE ended INT(1) ;
	DECLARE duplicate_handler INT(1) DEFAULT 0;
	DECLARE CONTINUE HANDLER FOR SQLSTATE '23000' SET duplicate_handler = 1;
	DECLARE EXIT HANDLER FOR SQLEXCEPTION
	BEGIN
		SELECT 'Error - terminating' AS result;
	END;
	INSERT INTO user_info(id) VALUES (v_id);
	IF (duplicate_handler = 1) THEN
		SELECT 'Duplicate Key Error' AS result;
	END IF;
END//
DELIMITER ;
```

DECLARE DUPLICATE_KEY CONDITION FOR 1062;
DECLARE CONTINUE HANDLER FOR DUPLICATE_KEY SET DUPLICATE = 1;
红色部分和紫色部分等价，一个是错误代码，一个是错误信息

触发器

- 特点

  1. 行级

  2. 临时行表(NEW,OLD)

  3. 前置？ 后置？(BEFORE, AFTER)

触发器

- 语法

  CREATE TRIGGER ***trigger_name***

  { BEFORE | AFTER }

  { INSERT | UPDATE | DELETE }

  ON table_name

  FOR EACH ROW

  triggered_statement

- 示例

  CREATE TRIGGER City_AD *AFTER DELETE* ON City

  FOR EACH ROW

  INSERT INTO DeletedCity (ID, Name) VALUES (OLD.ID, OLD.Name);

视图

- 特点

  虚拟表！

  可以基于磁盘表或者视图！

  可以更新？

- 优点

  数据安全?

  便于管理

视图–示例

```
CREATE
    ALGORITHM = 【MERGE|TEMPTABLE】
    VIEW `test`.`ha_user1`
    AS
    (SELECT user,password,host FROM mysql.user);
```

视图–限制

- 不包括子查询
- 不能在SELECT语句中包含系统变量或者临时变量
- 不包含动态语句处理
- 基表必须存在
- 基表必须是磁盘表
- 不能基于视图创建触发器

MySQL 开发与优化

谢谢！