益海嘉里MySQL读阻塞写问题排查

马飞

2018-5-29

1. 测试环境

数据库版本：MySQL 5.7.18

数据库隔离级别：REPEATABLE-READ

初始化数据：

|  |
| --- |
| CREATE TABLE xs(xh INT AUTO\_INCREMENT PRIMARY KEY,NAME VARCHAR(20),age INT,class INT);  INSERT INTO xs(NAME,age,class) VALUES('zhang.san',23,1);  INSERT INTO xs(NAME,age,class) VALUES('li.shi',28,2);  INSERT INTO xs(NAME,age,class) VALUES('wang.wu',38,1);  INSERT INTO xs(NAME,age,class) VALUES('zhao.liu',41,2);  CREATE TABLE class(id INT AUTO\_INCREMENT PRIMARY KEY,NAME VARCHAR(20));  INSERT INTO class(NAME) VALUES('A');  INSERT INTO class(NAME) VALUES('B'); |

1. 读操作是否阻塞写操作

## 2.1 自动提交模式

### 2.1.1 隔离级别：REPEATABLE-READ

会话1：

|  |
| --- |
| *#step:1*  SHOW VARIABLES LIKE '%iso%';  SHOW VARIABLES LIKE '%autocommit%';  *#step:2*  DELIMITER $$  USE `test`$$  DROP PROCEDURE IF EXISTS `proc\_xs\_sele`$$  CREATE DEFINER=`root`@`192.168.88.39` PROCEDURE `proc\_xs\_sele`(p\_times INT )  BEGIN  DECLARE n\_times INT;  DECLARE n\_rec INT DEFAULT 0;  SET n\_times=p\_times;  WHILE n\_times>0 DO  SELECT COUNT(0) INTO n\_rec FROM xs a,class b WHERE a.`class`=b.`id`;  SET n\_times=n\_times-1;  END WHILE;  END$$  DELIMITER ;  *#step:3 模拟一直有查询在进行*  CALL proc\_xs\_sele(100000000); |

会话2：

|  |
| --- |
| *#step:1*  SHOW VARIABLES LIKE '%iso%';  SHOW VARIABLES LIKE '%autocommit%';  *#step:2,在会话1，step3的执行过程中，反复执行下面的更新语句，均可正常执行。*  UPDATE xs SET age=age+1; |

**结论：**在自动提交模式下，隔离级别为可重复读级别下，读不会阻塞写操作。

### 2.1.2 隔离级别：READ-COMMITTED

会话1：

|  |
| --- |
| *#step:1*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  SHOW VARIABLES LIKE '%iso%';  SHOW VARIABLES LIKE '%autocommit%';  *#step:2*  SHOW VARIABLES LIKE '%iso%';  SHOW VARIABLES LIKE '%autocommit%';  *#step:3*  CALL proc\_xs\_sele(100000000); |

会话2：

|  |
| --- |
| *#step:1*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  *#step:2*  SHOW VARIABLES LIKE '%iso%';  SHOW VARIABLES LIKE '%autocommit%';  *#step:3在会话1，step3的执行过程中，反复执行下面的更新语句，均可正常执行。*  UPDATE xs SET age=age+1; |

**结论：**在自动提交模式下，隔离级别为已提交读级别下，读不会阻塞写操作。

## 2.2 非自动提交模式

### 2.2.1 隔离级别：REPEATABLE-READ

会话1：

|  |
| --- |
| *#step:1*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ ;  SET SESSION autocommit=0;  *#step:2*  SHOW VARIABLES LIKE '%iso%';  SHOW VARIABLES LIKE '%autocommit%';  *#step:3*  DELIMITER $$  USE `test`$$  DROP PROCEDURE IF EXISTS `proc\_xs\_sele`$$  CREATE DEFINER=`root`@`192.168.88.39` PROCEDURE `proc\_xs\_sele`(p\_times INT )  BEGIN  DECLARE n\_times INT;  DECLARE n\_rec INT DEFAULT 0;  SET n\_times=p\_times;  WHILE n\_times>0 DO  SELECT COUNT(0) INTO n\_rec FROM xs a,class b WHERE a.`class`=b.`id`;  SET n\_times=n\_times-1;  END WHILE;  END$$  DELIMITER ;  *#step:4 模拟一直有查询在进行* START TRANSACTION;  CALL proc\_xs\_sele(100000000); |

会话2：

|  |
| --- |
| *#step:1*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ ;  SET SESSION autocommit=0;  *#step2*  SHOW VARIABLES LIKE '%iso%';  SHOW VARIABLES LIKE '%autocommit%';  *#step:2,在“会话1”的step3的执行过程中，反复执行下面的更新语句，均可正常执行。*  START TRANSACTION;  UPDATE xs SET age=age+1; |

会话3：

|  |
| --- |
| *#step:1*  Select \* from xs; #此时查询出来的结果是会话2开启事务之前的结果，此时“会话1”仍在执行中  *#step2:查询“会话1”正在执行的语句*  SELECT USER,HOST,db,command,TIME,state,info FROM information\_schema.`PROCESSLIST` WHERE info LIKE '%xs%'; |

**结论：**在非自动提交模式下，隔离级别为可重复读级别下，读不会阻塞写操作。

### 2.2.2 隔离级别：READ-COMMITTED

会话1：

|  |
| --- |
| *#step:1*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  SET SESSION autocommit=0;  *#step:2*  SHOW VARIABLES LIKE '%iso%';  SHOW VARIABLES LIKE '%autocommit%';  *#step:3*  DELIMITER $$  USE `test`$$  DROP PROCEDURE IF EXISTS `proc\_xs\_sele`$$  CREATE DEFINER=`root`@`192.168.88.39` PROCEDURE `proc\_xs\_sele`(p\_times INT )  BEGIN  DECLARE n\_times INT;  DECLARE n\_rec INT DEFAULT 0;  SET n\_times=p\_times;  WHILE n\_times>0 DO  SELECT COUNT(0) INTO n\_rec FROM xs a,class b WHERE a.`class`=b.`id`;  SET n\_times=n\_times-1;  END WHILE;  END$$  DELIMITER ;  *#step:4 模拟一直有查询在进行* START TRANSACTION;  CALL proc\_xs\_sele(100000000); |

会话2：

|  |
| --- |
| *#step:1*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  SET SESSION autocommit=0;  *#step2*  SHOW VARIABLES LIKE '%iso%';  SHOW VARIABLES LIKE '%autocommit%';  *#step:2,在“会话1”的step3的执行过程中，反复执行下面的更新语句，均可正常执行。*  START TRANSACTION;  UPDATE xs SET age=age+1; |

会话3：

|  |
| --- |
| *#step:1*  Select \* from xs; #此时查询出来的结果是会话2开启事务之前的结果，此时“会话1”仍在执行中  *#step2:查询“会话1”正在执行的语句*  SELECT USER,HOST,db,command,TIME,state,info FROM information\_schema.`PROCESSLIST` WHERE info LIKE '%xs%'; |

**结论：**在非自动提交模式下，隔离级别为已提交读级别下，读不会阻塞写操作。