CEDAR 测试案例

修订历史

版本	修订日期	修订描述	作者	备注
CEDAR 0.2	**2016-09-25**	**CEDAR 测试案例**	**李捷荧**	

表锁

编号	1	配置	10.11.1.203 单机集群
测试目的	单事务单表加锁后进行增删改查操作，看功能是否正常		

测试输入	START TRANSACTION;
	LOCK TABLE t1;
	SELECT * FROM t1;
	COMMIT;
	
	START TRANSACTION;
	LOCK TABLE t1;
	insert INTO t1 VALUES(7, 1, '2003-1-1');
	COMMIT;
	
	START TRANSACTION;
	LOCK TABLE t1;
	Update t1 SET value2= 'adc' WHERE id=1;
	COMMIT;
	
	START TRANSACTION;
	LOCK TABLE t1;
	Delete from t1 where id=7;
	COMMIT;		

编号	2	配置	10.11.1.203 单机集群
测试目的	当一个事务对一张表进行查询操作，在事务结束之前，用其他事务对该表进行加锁操作		

测试输入	**Client1:**
	START TRANSACTION;
	SELECT * FROM t1;
	Client2：
	START TRANSACTION;
	LOCK TABLE t1;
	UPDATE t1 SET value2=22222 WHERE id=1;		

	COMMIT; COMMIT; 查询、锁表、更新成功		
编号	3	配置	10.11.1.203 单机集群
测试目的	当一个事务对一张表进行更新操作，在事务结束之前，用其他事务对该表进行加锁操作		

测试输入	**Client1:**
	START TRANSACTION;
	INSERT INTO t1 VALUES(4, 4,'ee');
	Client2：
	START TRANSACTION;
	LOCK TABLE t1;
	UPDATE t1 SET value1=222 WHERE id=1;
	COMMIT;
	COMMIT;
	插入、更新成功，锁表失败
	
	Client1:
	START TRANSACTION;
	Client2：
	START TRANSACTION;
	LOCK TABLE t1;
	UPDATE t1 SET value2=111 WHERE id=1;
	UPDATE t1 SET value1=222 WHERE id=4;
	COMMIT;
	COMMIT;
	Client1 锁表成功，更新成功，**Client2** 更新失败
	
	Client1:
	START TRANSACTION;
	UPDATE t1 SET value1=11111 WHERE id=1;
	Client2：
	START TRANSACTION;
	LOCK TABLE t1;
	UPDATE t1 SET value1=3333 WHERE id=4;
	COMMIT;
	COMMIT;
	Client1 更新成功，**Client2** 锁表失败，更新成功
	
	Client1:
	START TRANSACTION;
	DELETE FROM t1 WHERE id=4;
	Client2：
	START TRANSACTION;		

	LOCK TABLE t1;
	UPDATE t1 SET value1=11 WHERE id=1;
	COMMIT;
	COMMIT;
	Client1 删除成功，**Client2** 锁表失败，更新成功		

编号	4	配置	10.11.1.203 单机集群
测试目的	当一个事务对一张表进行加锁后查询操作，在事务结束之前，用其他事务对该表进行更新操作。		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	Select * FROM t1;
	Client2：
	START TRANSACTION;
	UPDATE t1 SET value2=11 WHERE id=1;
	COMMIT;
	COMMIT;
	Client1 锁表成功，查询成功，**Client2** 更新失败		

编号	5	配置	10.11.1.203 单机集群
测试目的	当一个事务对一张表进行加锁后进行更新操作，在事务结束之前，用其他事务对该表进行更新操作		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	INSERT INTO t1 VALUES(6, 6,'FF');
	Client2：
	START TRANSACTION;
	UPDATE t1 SET value1=66 WHERE id=6;
	COMMIT;
	COMMIT;
	Client1 锁表成功，插入成功，**Client2** 更新失败
	
	Client1:
	START TRANSACTION;
	LOCK TABLE t1;
	UPDATE t1 SET value2=66 WHERE id=6;
	Client2：
	START TRANSACTION;
	UPDATE t1 SET value1=131 WHERE id=3;
	COMMIT;
	COMMIT;
	Client1 锁表成功，更新成功，**Client2** 更新失败		

	Client1:
	START TRANSACTION;
	LOCK TABLE t1;
	DELETE FROM t1 WHERE id=6;
	Client2：
	START TRANSACTION;
	UPDATE t1 SET value1=22 WHERE id=1;
	SELECT * FROM t1;
	COMMIT;
	COMMIT;
	Client1 锁表成功，删除成功，**Client2** 更新失败，查询成功		

编号	6	配置	10.11.1.203 单机集群
测试目的	当一个事务对一张表进行加锁后查询，在事务结束之后，用其他事务对该表进行更新操作。		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	Select * FROM t1;
	COMMIT;
	
	Client2：
	START TRANSACTION;
	UPDATE t1 SET value1=141 WHERE id=1;
	COMMIT;
	Client1 锁表成功，查询成功，**Client2** 更新成功		

编号	7	配置	10.11.1.203 单机集群
测试目的	当一个事务对一张表进行加锁后进行更新操作，在事务结束之后，用其他事务对该表进行更新操作		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	INSERT INTO t1 VALUES(3, 6,'EE');
	COMMIT;
	Client2：
	START TRANSACTION;
	UPDATE t1 SET value2=666 WHERE id=3;
	COMMIT;
	Client1 锁表成功，插入成功，**Client2** 更新成功
	
	Client1:
	START TRANSACTION;
	LOCK TABLE t1;
	UPDATE t1 SET value1=1441 WHERE id=3;
	COMMIT;		

	Client2：
	START TRANSACTION;
	UPDATE t1 SET　value2=111 WHERE id=3;
	COMMIT;
	Client1 锁表成功，更新成功，**Client2** 更新成功
	
	Client1:
	START TRANSACTION;
	LOCK TABLE t1;
	DELETE FROM t1 WHERE id =3;
	COMMIT;
	Client2：
	START TRANSACTION;
	UPDATE t1 SET　value1=22 WHERE id=1;
	COMMIT;
	Client1 锁表成功，删除成功，**Client2** 更新成功		

编号	8	配置	10.11.1.203 单机集群
测试目的	当一个事务对一张表进行加锁操作，在事务结束之前，用其他事务对该表进行加锁操作		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	Client2：
	START TRANSACTION;
	LOCK TABLE t1;
	COMMIT;
	COMMIT;
	Client1 锁表成功，**Client2** 锁表失败		

编号	9	配置	10.11.1.203 单机集群
测试目的	当一个事务对一张表进行加锁操作，在事务结束之后，用其他事务对该表进行加锁操作		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	COMMIT;
	Client2：
	START TRANSACTION;
	LOCK TABLE t1;
	COMMIT;
	Client1 锁表成功，**Client2** 锁表成功		

编号	10	配置	10.11.1.203 单机集群

测试目的	当一个事务对多张表进行加锁操作，在事务结束之前，用其他事务对其中的表进行加锁修改操作		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	LOCK TABLE t2;
	Client2：
	START TRANSACTION;
	LOCK TABLE t1;
	UPDATE t2 SET value2=1234 WHERE id=1;
	SELECT * FROM t1;
	COMMIT;
	COMMIT;
	Client1 锁表成功，**Client2** 锁表失败，更新失败		

编号	11	配置	10.11.1.203 单机集群
测试目的	当一个事务对多张表进行加锁操作，在事务结束之前加锁过程中，用其他事务对其中的表进行加锁修改操作		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	Insert into t1 values(2,1, 'bb');
	UPDATE t1 SET value2=89 where id=1;
	SELECT * FROM t1;
	Client2：
	START TRANSACTION;
	LOCK TABLE t2;
	LOCK TABLE t2;
	INSERT INTO t2 VALUES(5,3, 'bb');
	UPDATE t2 SET value2=478 WHERE id=3;
	SELECT * FROM t2;
	COMMIT;
	COMMIT;
	Client1：t1 表锁表成功，插入成功，查询成功。t2 表锁表失败，插入失败，查询成功。
	Client2：t2 表锁表成功，插入成功		

编号	12	配置	10.11.1.203 单机集群
测试目的	不在事务中锁定一张表		
测试输入	LOCK TABLE t1;		
编号	13	配置	10.11.1.203，10.11.1.201,10.11.1.204 三机集群
测试目的	在一个事务中锁定一张表，ups 换主后，在另一个事务中锁定同一张表		
测试输入	**Client1:** START TRANSACTION;		

	LOCK TABLE t1;
	UPS 换主
	Client2：
	START TRANSACTION;
	LOCK TABLE t1;
	COMMIT;
	COMMIT;		

编号	14	配置	10.11.1.201 单集群
测试目的	在事务内进行锁表操作后，锁定不存在的表		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	insert into t1 values(3,2,'a');
	LOCK TABLE t11;
	insert into t1 values(4,2,'a');
	COMMIT;
	Client1: t1 表锁表成功，插入成功，t11 表锁定失败		

编号	15	配置	10.11.1.201 单集群
测试目的	在事务内进行锁表操作，错误更新语句，		

测试输入	**Client1:**
	START TRANSACTION;
	insert into t1 values(5,2,'a');
	LOCK TABLE t1;
	update t1 set value1='aa' where id=1;
	commit;
	Client1: t1 表锁表成功，插入成功，更新失败		

编号	16	配置	10.11.1.201 单集群
测试目的	在事务内进行锁表操作，错误锁表语句		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	insert into t1 values(6,2,'a');
	loc table t1;
	insert into t1 values(7,2,'aa');
	commit;
	Client1: t1 表锁表失败，插入成功		

编号	17	配置	10.11.1.201 单集群
测试目的	在事务内进行锁表操作，插入主键相同的语句		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	insert into t1 values(12,2,'aa');
	insert into t1 values(10,3,'aa');		

	insert into t1 values(13,2,'aa');
	commit;
	Client1: t1 表锁表成功，插入(12,2,'aa') (13,2,'aa')成功（10 主键重复）		

编号	18	配置	10.11.1.201 单集群
测试目的	在事务内锁定不存在的表		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t11;
	insert into t1 values(14,2,'aa');
	commit;
	Client1: t11 表锁表失败，t1 表插入成功		

编号	19	配置	10.11.1.201 单集群
测试目的	在事务内进行错误更新语句，		

测试输入	**Client1:**
	START TRANSACTION;
	update t1 set value1= 'hhhh' ,value2= 'hhhh' where id=1;
	insert into t1 values(18,2,'accsvv');
	commit;
	Client1: 更新时将 value1 当作 0 更新，插入成功		

编号	20	配置	10.11.1.201 单集群
测试目的	在事务内进行错误锁表语句		

测试输入	**Client1:**
	START TRANSACTION;
	insert into t1 values(19,2,'aa');
	loc table t1;
	insert into t1 values(20,2,'aa');
	Commit;
	Client1: t1 表锁表失败，插入成功		

编号	21	配置	10.11.1.201 单集群
测试目的	在事务内插入主键相同的语句		

测试输入	**Client1:**
	START TRANSACTION;
	insert into t1 values(21,2,'aa');
	insert into t1 values(21,3,'aa');
	insert into t1 values(22,2,'aa');
	commit;
	Client1: t1 表插入(21,2,'aa') (22,2,'aa')成功		

编号	22	配置	10.11.1.201 单集群
测试目的	在事务内进行锁表操作，先对该表进行查询操作，再对该表进行大量的更新操作，等事务回滚后，再对该表进行查询操作		

测试输入	**Client1:**
	START TRANSACTION;
	SELECT * FROM t1;
	LOCK TABLE t1;		

	Replace into t1 (id,value1,value2) values(?,?,?);
	SELECT * FROM t1 limit 5;
	锁表成功，事务回滚，插入失败
	
	Client1:
	START TRANSACTION;
	SELECT * FROM t1;
	LOCK TABLE t1;
	update t1 set value1=?,value2=? Where id=?;
	SELECT * FROM t1 limit 5;
	锁表成功，事务回滚，更新失败
	
	Client1:
	START TRANSACTION;
	SELECT * FROM t1;
	LOCK TABLE t1;
	Insert into t1 values(?,?,?);
	SELECT * FROM t1 limit 5;
	锁表成功，事务回滚，插入失败
	
	START TRANSACTION;
	SELECT * FROM t1;
	LOCK TABLE t1;
	delete from t1 where id=?;
	SELECT * FROM t1 limit 5;
	锁表成功，事务回滚，删除失败		

编号	23	配置	10.11.1.201 单集群
测试目的	在事务内进行锁表操作，先对该表进行查询操作，再对该表进行大量的更新操作，结束事务后，再对该表进行查询操作		

测试输入	**Client1:**
	START TRANSACTION;
	SELECT * FROM t1;
	LOCK TABLE t1;
	Replace into t1 (id,value1,value2) values(?,?,?);
	COMMIT;
	SELECT * FROM t1 limit 5;
	锁表成功，插入成功
	
	Client1:
	START TRANSACTION;
	SELECT * FROM t1;
	LOCK TABLE t1;
	update t1 set value1=?,value2=? Where id=?,?,?;
	COMMIT;		

	SELECT * FROM t1 limit 5;
	锁表成功，更新成功
	
	Client1:
	START TRANSACTION;
	SELECT * FROM t1;
	LOCK TABLE t1;
	Insert into t1 values(?,?,?);
	COMMIT;
	SELECT * FROM t1 limit 5;
	锁表成功，插入成功
	
	Client1:
	START TRANSACTION;
	SELECT * FROM t1;
	LOCK TABLE t1;
	delete from t1 where id=?;
	COMMIT;
	SELECT * FROM t1 limit 5;
	锁表成功，删除成功		

编号	24	配置	10.11.1.201 单集群
测试目的	在事务中进行大量的锁表操作，看锁表操作是否成功。		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE tn;
	COMMIT;		

编号	25	配置	10.11.1.201 单集群
测试目的	在事务 1 内进行锁表操作，在事务 1commit 之前，在其他事务中对该表进行大量更新操作,看更新是否成功。		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	Client2：
	START TRANSACTION;
	Replace into t1 (id,value1,value2) values(?,?,?);
	COMMIT;
	COMMIT;
	Client1:锁表成功
	Client2:更新失败
	
	Client1:
	STA2RT TRANSACTION;
	LOCK TABLE t1;
	Client2：		

	START TRANSACTION;
	Insert into t1 values(?,?,?);
	COMMIT;
	COMMIT;
	Client1:锁表成功
	Client2:插入失败
	
	Client1:
	START TRANSACTION;
	LOCK TABLE t1;
	Client2：
	START TRANSACTION;
	update t1 set value1=?,value2=? Where id=?,?,?;
	COMMIT;
	COMMIT;
	Client1:锁表成功
	Client2:更新失败
	
	Client1:
	START TRANSACTION;
	LOCK TABLE t1;
	Client2：
	START TRANSACTION;
	delete from t1 where id=?;
	COMMIT;
	COMMIT;
	Client1:锁表成功
	Client2:删除失败		

编号	26	配置	10.11.1.201 单集群
测试目的	在事务 1 内进行锁表操作，在事务 1commit 之前，在其他事务中对该表进行大量锁表操作,看锁表是否成功。		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	Client2：
	START TRANSACTION;
	LOCK TABLE t1; ……
	COMMIT;
	COMMIT;
	Client1:锁表成功
	Client2: 锁表失败		

编号	27	配置	10.11.1.201 单集群
测试目的	在事务 1 内对 t1 表进行大量更新操作，在事务 1commit 之前，在其他事务中重复对该表加锁,看更新是否成功，锁表是否成功。		

测试输入	**Client1:**
	START TRANSACTION;
	replace into t1 (id,value1,value2) values(?,?,?);
	Client2：
	START TRANSACTION;
	LOCK TABLE t1; ……
	COMMIT;
	COMMIT;
	Client1:replace 成功
	Client2: 锁表失败
	
	Client1:
	START TRANSACTION;
	insert into t1 values(?,?,?);
	Client2：
	START TRANSACTION;
	LOCK TABLE t1; …….
	COMMIT;
	COMMIT;
	Client1: 插入成功
	Client2: 锁表失败
	
	Client1:
	START TRANSACTION;
	update t1 set value1=?,value2=? Where id=?,?,?;
	Client2：
	START TRANSACTION;
	LOCK TABLE t1; …….
	COMMIT;
	COMMIT;
	Client1: 更新成功
	Client2: 锁表失败
	
	Client1:
	START TRANSACTION;
	delete from t1 where id=?;
	Client2：
	START TRANSACTION;
	LOCK TABLE t1; …….
	COMMIT;
	COMMIT;
	Client1: 删除成功
	Client2: 锁表失败		

编号	28	配置	10.11.1.201 10.11.1.203 10.11.1.204 三集群

测试目的	在事务 1 锁定 t1 表后，强制换主，看 t1 表是否被释放		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	
	强制换主
	
	Client2：
	START TRANSACTION;
	LOCK TABLE t1;
	COMMIT;
	COMMIT;
	
	Client2 锁表失败显示 t1 表已被锁定		

编号	29	配置	10.11.1.201 10.11.1.203 10.11.1.204 三集群
测试目的	在事务 1 锁定 t1 表后，每日合并，看 t1 表是否被释放		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	
	每日合并
	
	Client2：
	START TRANSACTION;
	LOCK TABLE t1;
	COMMIT;
	COMMIT;
	T1 表被及时释放		

编号	32	配置	10.11.1.201 10.11.1.203 10.11.1.204 三集群 (201 为主)
测试目的	在 201 上对 t1 表进行大量的更新操作，在 203 或 204 上开启事务 1 锁定 t1 表后，在 203 或 204 上查询		

测试输入	replace into t1 (id,value1,value2) values(?,?,?);
	
	Client1:
	START TRANSACTION;
	LOCK TABLE t1;
	select count(*) from t1;
	commit;
	commit;
	Client2：
	select count(*) from t1;
	
	201 有时开启不了新链接，203、204 可以开启新链接		

	事务 1 中查询成功，commit 成功 事务 2 中查询成功		
编号	34	配置	10.11.1.201 10.11.1.203 10.11.1.204 三集群
测试目的	在事务 1 锁定 t1 表后，开启新事务 replace t1 表，在事务 1commit 后，看新事务是否执行成功		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;
	
	Client2：
	START TRANSACTION;
	replace into t1 (id,value1,value2) values(?,?,?);
	
	COMMIT;
	COMMIT;
	
	Client1 未 commit 之前 **Client2** replace 失败
	Client1 commit 之后 **Client2** replace 成功		

编号	35	配置	10.11.1.201 10.11.1.203 10.11.1.204 三集群
测试目的	一边大量 replace，一边 lock table		

测试输入	**Client1:**
	START TRANSACTION;
	LOCK TABLE t1;…….
	COMMIT;
	Client2：
	START TRANSACTION;
	replace into t1 (id,value1,value2) values(?,?,?);
	COMMIT;		

编号	36	配置	10.11.1.201 10.11.1.203 10.11.1.204 三集群
测试目的	在事务中对 t1 表加排它锁后，进行集群换主		

测试输入	**Client1:**
	START TRANSACTION;
	Select * from t1 where id=1 for update;
	
	集群换主
	
	Client2:
	START TRANSACTION;
	Select * from t1 where id=1 for update;
	
	T1 表被锁定		

Bloom-filter Join

编号	1	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Bloomfilter_joinge 在都为小表无区分 1W 和 50W 情况下两表连接时各连接类型（left,inner, right, full outer）功能是否正常		
测试输入	Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7; Pass 10010 1.29 sec Select /*+ join(bloomfilter_join) */ count(*) from obst1 inner join obst2 on obst1.obstcol7 = obst2.obstcol7; Pass 428 1.27 sec Select /*+ join(bloomfilter_join) */ count(*) from obst1 right join obst2 on obst1.obstcol7 = obst2.obstcol7; Pass 500000 1.68 sec Select /*+ join(bloomfilter_join) */ count(*) from obst1 full outer join obst2 on obst1.obstcol7 = obst2.obstcol7; Pass 509582 1.68 sec		
编号	2	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Bloomfilter_join 在都为小表无区分 1W 和 50W 情况下两表连接各相同连接列类型（int, varchar, double, bool, timestamp, decimal）功能是否正常		

测试输入	Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7;
	Pass 10010 1.27 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol2 = obst2.obstcol2;
	Pass 10000 1.85 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol3 = obst2.obstcol3;
	Pass 10007 1.48 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst4 left join obst2 on obst4.obstcol4 = obst2.obstcol4;
	Pass 5000000000 1 hour 21 min 15.17 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol5 = obst2.obstcol5;		

	Pass 10013 1.73 sec Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol6 = obst2.obstcol6; Pass 38762803 41.99 sec		
编号	3	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Bloomfilter_join 在都为小表无区分 1W 和 50W 情况下两表连接各不同连接列类型（int, varchar, double, bool, timestamp, decimal）功能是否正常		
测试输入	int /double Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol3; ERROR 10009 1.90 sec int/decimal Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7= obst2.obstcol6; Pass 10000 1.14sec double/decimal Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol3 = obst2.obstcol6; ERROR 10000 1.69sec		
编号	4	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Bloomfilter_join 在大小表区分明显 1W 和 1000W 情况下，两表连接各相同连接列类型（int, varchar, double, bool, timestamp, decimal）功能是否正常		

测试输入	Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst6 on obst1.obstcol7 = obst6.obstcol7;
	Pass 12678 53.12 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst6 on obst1.obstcol2 = obst6.obstcol2;
	Pass 10000 2 min 13.10 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst6 on obst1.obstcol3 = obst6.obstcol3;
	Pass 13170 2 min 22.46 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst6 on obst1.obstcol4= obst6.obstcol4;
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst6 on obst1.obstcol5= obst6.obstcol5;		

	Pass 13169 1 min 59.20 sec Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst6 on obst1.obstcol6= obst6.obstcol6; Pass 775195892 15 min 0.23 sec		
编号	5	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Bloomfilter_join 在都为大表无区分 500W 和 1000W 情况下，两表连接各相同连接列类型（int, varchar, double, bool, timestamp, decimal）功能是否正常		

测试输入	Select /*+ join(bloomfilter_join) */ count(*) from obst5 left join obst6 on obst5.obstcol7= obst6.obstcol7;
	Pass 6562812 3 min 33.15 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst5 left join obst6 on obst5.obstcol2= obst6.obstcol2;
	Pass 5000037 3 min 8.04 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst5 left join obst6 on obst5.obstcol3= obst6.obstcol3;
	Pass 6563351 3 min 0.69 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst5 left join obst6 on obst5.obstcol4= obst6.obstcol4;
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst5 left join obst6 on obst5.obstcol5= obst6.obstcol5;
	Pass 6562636 3 min 18.69 sec
	
	Select /*+ join(bloomfilter_join) */ count(*) from obst5 left join obst6 on obst5.obstcol6= obst6.obstcol6;		

编号	6	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Bloomfilter_joinge 多表连接各连接类型（left,inner, right, full outer）单独使用功能是否正常		
测试输入	Obst1 数据量 1w obst2 数据量 50w obst3 数据量 500w Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 left join obst3 on obst2.obstcol7 = obst3.obstcol7; Pass 10048 53.76 sec Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 inner join obst2 on obst1.obstcol7 = obst2.obstcol7 inner join obst3 on obst2.obstcol7 = obst3.obstcol7; Pass 189 58.39 sec		

	Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 right join obst2 on obst1.obstcol7 = obst2.obstcol7 right join obst3 on obst2.obstcol7 = obst3.obstcol7; Pass 5005100 1 min 4.20 sec Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 full outer join obst2 on obst1.obstcol7 = obst2.obstcol7 full outer join obst3 on obst2.obstcol7 = obst3.obstcol7; Pass 5332081 59.72 sec		
编号	7	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Bloomfilter_joinge 多表连接各连接类型（left,inner, right, full outer）混合使用功能是否正常		
测试输入	Obst1 数据量 1w obst2 数据量 50w obst3 数据量 500w left/inner Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 inner join obst3 on obst2.obstcol7 = obst3.obstcol7; Pass 189 1.62 sec left/right Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 right join obst3 on obst2.obstcol7 = obst3.obstcol7; Pass 5000005 1 min 1.15 sec left/full outer Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 full outer join obst3 on obst2.obstcol7 = obst3.obstcol7; Pass 500986 1 min 14.30 sec inner/right Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 inner join obst2 on obst1.obstcol7 = obst2.obstcol7 right join obst3 on obst2.obstcol7 = obst3.obstcol7; Pass 10048 58.00 sec inner/full outer Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 inner join obst2 on obst1.obstcol7 = obst2.obstcol7 full outer join obst3 on obst2.obstcol7 = obst3.obstcol7; Pass 501244 1 min 21.30 sec		

	right/full outer Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 right join obst2 on obst1.obstcol7 = obst2.obstcol7 full outer join obst3 on obst2.obstcol7 = obst3.obstcol7; Pass 571201 1 min34.30 sec		
编号	8	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Bloomfilter_join 多表连接各相同连接列类型（int, varchar, double, bool, timestamp, decimal）功能是否正常		

测试输入	Obst1 数据量 1w obst2 数据量 50w obst3 数据量 500w
	
	Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 left join obst3 on obst2.obstcol7 = obst3.obstcol7;
	Pass 10048 58.00 sec
	
	Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol2 = obst2.obstcol2 inner join obst3 on obst2.obstcol2 = obst3.obstcol2;
	Pass 10000 1 min 9.70 sec
	
	Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol3 = obst2.obstcol3 left join obst3 on obst2.obstcol3 = obst3.obstcol3;
	Pass 10040 59.18 sec
	
	Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol4 = obst2.obstcol4 inner join obst3 on obst2.obstcol4 = obst3.obstcol4;
	
	Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol5 = obst2.obstcol5 right join obst3 on obst2.obstcol5 = obst3.obstcol5;
	Pass 10059 58.69 sec
	
	Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol6 = obst2.obstcol6 inner join obst3 on obst2.obstcol6 = obst3.obstcol6;
	Pass 0 25 min 46.16 sec		

编号	9	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Bloomfilter_join 多表连接各不同连接列类型（int, varchar, double, bool, timestamp, decimal）功能是否正常		
测试输入	Obst1 数据量 1w obst2 数据量 50w obst3 数据量 500w		

	int/double Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol3 left join obst3 on obst2.obstcol7 = obst3.obstcol3; ERROR 44176 3.45 sec Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 left join obst3 on obst2.obstcol3 = obst3.obstcol3; Pass 44024 3.44s int/decimal Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol6 left join obst3 on obst2.obstcol7 = obst3.obstcol6; Pass 91646 3.28s Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 left join obst3 on obst2.obstcol6 = obst3.obstcol6; Pass 44024 3.45s double/decimal Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol3 = obst2.obstcol6 left join obst3 on obst2.obstcol3 = obst3.obstcol6; ERROR 72023 3.74s Select /*+ join(bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol3 = obst2.obstcol3 left join obst3 on obst2.obstcol6 = obst3.obstcol6; Pass 43609 3.49s int/double/decimal Select /*+ join(bloomfilter_join，bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 inner join obst3 on obst2.obstcol3 = obst3.obstcol3 left join obst4 on obst2.obstcol6 = obst4.obstcol6 ; ERROR 737982 53.69 sec Select /*+ join(bloomfilter_join，bloomfilter_join，bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol3 inner join obst3 on obst2.obstcol7 = obst3.obstcol6 left join obst4 on obst2.obstcol3 = obst4.obstcol6 ; Pass 10009 54.57 sec		
编号	10	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Merge_join 与 Bloomfilter_join 混合多表连接（left/inner/right/full outer）功能是否正常		

测试输入	Obst1 数据量 1w obst2 数据量 50w obst3 数据量 500w obst4 数据量 50w obst5 数据量 500w Select /*+ join(bloomfilter_join,merge_join,bloomfilter_join,merge_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 right join obst3 on obst2.obstcol7 = obst3.obstcol7 inner join obst4 on obst3.obstcol7 = obst4.obstcol7 full outer join obst5 on obst4.obstcol7 = obst5.obstcol7 ; PASS 5166522 3 min 55.22 sec		
编号	11	配置	197，单集群，RS+UPS+CS+MS
测试目的	测试 Merge_join 与 Bloomfilter_join 混合多表连接时，根据 join 类型数与数据表数的关系不对应时，功能是否正常		
测试输入	Obst1 数据量 1w obst2 数据量 50w obst3 数据量 500w join 类型数 ＞ 数据表数 Select /*+ join(bloomfilter_join,merge_join,bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 inner join obst3 on obst2.obstcol7 = obst3.obstcol7; PASS 189 1 min 2.79 sec 当 join 类型数 ＜ 数据表数 Select /*+ join(bloomfilter_join,merge_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7 inner join obst3 on obst2.obstcol7 = obst3.obstcol7 inner join obst4 on obst3.obstcol7 = obst4.obstcol7; PASS 8 47.63 sec		
编号	12	配置	197，198，199，三集群，RS+UPS+CS+MS
测试目的	测试在每日合并期间，Bloomfilter_join 功能是否正常		
测试输入	Obst1 数据量 1w obst2 数据量 50w 每日合并期间 Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7; PASS		
编号	13	配置	197，198，199，三集群，RS+UPS+CS+MS
测试目的	测试在集群换主期间，Bloomfilter_join 功能是否正常		
测试输入	Obst1 数据量 1w obst2 数据量 50w obst3 集群换主期间 Select /*+ join(bloomfilter_join) */ count(*) from obst1 left join obst2 on obst1.obstcol7 = obst2.obstcol7; PASS		

快照读隔离级别

编号	1	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192：RS、UPS、MS、CS
测试目的	在三集群下，ReadCommitted 和 RepeatableRead 的非事务 select 查询是否正确		

测试输入	create table t1 (c1 int primary key, c2 int);
	insert into t1 values(1,1),(2,2),(3,3);
	set session tx_isolation=’READ-COMMITTED’;
	select * from t1;
	set session tx_isolation=’REPEATABLE-READ’;
	select * from t1;		

编号	2	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192：RS、UPS、MS、CS
测试目的	三集群，RepeatableRead 能否防脏读和不可重复读		

测试输入	create table t1 (c1 int primary key, c2 int);
	for(int i=1;i<=10;i++)
	   insert into t1 values(i,i);
	
	写事务会话 20 个，读事务会话 300 个
	写事务 i 里执行 1 次
	   update t1 set c2=c2+1 where c1=i;
	   insert into t1 values(i+10000,i+10000);
	   replace into t1 values(i+20000,i+20000);
	   delete from t1 where c1=i+10000);
	读事务里执行 3 次
	   select * from t1;
	   第一次在写事务更新操作执行前，第二次在更新操作执行后，第三次在写事务提交后		

编号	3	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192：RS、UPS、MS、CS
测试目的	三集群，切换隔离级别，读事务返回结果是否正确		
测试输入	create table t1 (c1 int primary key, c2 int); for(int i=1;i<=10;i++) insert into t1 values(i,i); 1. 全局 RepeatableRead，读写事务运行过程中，创建新会话将全局隔离级别切换为 ReadCommitted 2. 全局 ReadCommitted，读写事务执行过程中，创建新会话将全局隔离级别切换为 RepeatableRead 3. 全局 ReadCommitted，写事务先于读事务结束，读事务会话切		

	换为会话 RepeatableRead 4. 全局 RepeatableRead，写事务先于读事务结束，读事务会话切换为会话 ReadCommitted		
编号	4	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192：RS、UPS、MS、CS
测试目的	三集群，RepeatableRead 能否防止第一类和第二类丢失更新		

测试输入	create table t1(c1 int primary key, c2 int);
	for(int i=1;i<=10;i++)
	   insert into t1 values(i,i);
	
	第一类丢失更新
	   transactionA
	   update t1 set c2=c2+2 where c1=1;
	   commit
	   transactionB
	   update t1 set c2=c2-1 where c1=1;
	   rollback
	第二类丢失更新
	   transactionC
	   update t1 set c2=c2+1 where c1=1;
	   commit
	   transactionD
	   update t1 set c2=c2+1 where c1=1;
	   commit		

编号	5	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192：RS、UPS、MS、CS
测试目的	三集群，两种隔离级别下事务能否都读到自己修改但尚未提交的数据		

测试输入	create table t1(c1 int primary key, c2 int);
	for(int i=1;i<=10;i++)
	   insert into t1 values(i,i);
	
	ReadCommitted 下，readCommittedCount 个事务
	   for(int i=1;i<=readCommittedCount;i++)
	   {
	      update t1 set c2=c2+1 where c1=i;
	      select * from t1 where c1=i;
	   }
	RepeatableRead 下，repeatableReadCount 个事务
	   for(int j= readCommittedCount+1;j<repeatableReadCount;j++)
	   {
	      update t1 set c2=c2+1 where c1=j;		

	select * from t1 where c1=j;
	}		

编号	6	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192：RS、UPS、MS、CS
测试目的	三集群，换主之后 RepeatableRead 能否正常工作		

测试输入	原主集群为 190，现切换为 192
	
	create table t1(c1 int primary key, c2 int);
	for(int i=1;i<=10;i++)
	insert into t1 values(i,i);
	
	利用案例 2 测试
	利用案例 4 测试
	利用案例 5 测试		

编号	7	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192：RS、UPS、MS、CS
测试目的	三集群，故障恢复后 Repeatable 能否正常工作		

测试输入	create table t1(c1 int primary key, c2 int);
	for(int i=1;i<=10;i++)
	insert into t1 values(i,i);
	
	1. 杀掉一整个备集群再恢复备集群
	2. 杀掉主集群，备切主
	3. 杀掉所有集群再恢复		

编号	8	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192：RS、UPS、MS、CS
测试目的	三集群，RepeatableRead，当一份数据存在非常多的版本，读事务是否还可重复读		

测试输入	create table t1(c1 int primary key, c2 int);
	insert into t1 values(1,1),(2,2),(3,3);
	
	一个会话执行非事务的 10 万次单行更新 update t1 set c2=c2+1 where c1=1;
	另一个会话执行读事务，事务包含 10 万次 select c1,c2 from t1 where c1=1;		

编号	9	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192：RS、UPS、MS、CS
测试目的	三集群，RepeatableRead，当一份数据存在非常多的版本，执行每		

	日合并后，读事务是否还可重复读		

测试输入	create table t1(c1 int primary key, c2 int);
	insert into t1 values(1,1),(2,2),(3,3),….(10,10);
	
	一个会话执行非事务的 10 万次单行更新 update t1 set c2=c2+1 where c1=1;
	另一个会话执行读事务，事务包含 10 万次 select c1,c2 from t1 where c1=1;
	两个事务执行过程中执行每日合并		

编号	10	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192：RS、UPS、MS、CS
测试目的	三集群，RepeatableRead，更新事务执行过程中每日合并，rollback 结果是否正确		

测试输入	create table t1(c1 int primary key, c2 int);
	insert into t1 values(1,1),(2,2),(3,3),….(10,10);
	
	一个会话执行读事务，事务执行 3 万次：
	   update t1 set c2=c2+1 where c1=1;
	   select c1,c2 from t1 where c1=1;
	
	事务执行更新操作过程中执行每日合并（每日合并开始后事务的更新操作将导致事务 rollback）		

日志同步优化

编号	1	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192-198：RS、UPS、LMS + 6 * (MS + CS) 190、191 为备集群、193 为主集群 测试机：199
测试目的	测试在多次换主下，集群能否正常提供服务		
测试输入	 数据表 schema： drop table if exists reelecttest; create table reelecttest(id int, value varchar(20), primary key(id)); 周期性换主，并在换主的过程压有少量的负载（每秒几百条更新操作） 查看集群角色和换主的相关命令： ./rs_admin -r 10.11.1.191 -p 31500 get_obi_role ./rs_admin -r 10.11.1.191 -p 31500 reelect		

编号	2	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192-198：RS、UPS、LMS + 6 * (MS + CS) 190、191 为备集群、193 为主集群 测试机：199（3MS）、201、202、203
测试目的	测试在多次换主下，集群能否保证数据的一致性		

测试输入	
	负载 1：
	数据表 schema：
	drop table if exists reelecttest; create table reelecttest(id int, value varchar(20), primary key(id));
	
	周期性换主，并在换主的过程压有少量的负载（每秒几百条更新操作）
	
	负载 2：
	数据表 schema：
	drop table if exists t1;
	create table t1 (c1 int primary key, c2 int, c3 varchar(50));
	字符串的平均长度为 25
	
	向 OB 中插入数据（replace），100 * 6 个线程（主集群每个 MS 连接 100 个线程）
	共插入 1.2y 条数据
	
	检查数据表中数据是否有 1.2y 条
	set @@session.ob_query_timeout=9000000000;
	select count(*) from t1;		

编号	3	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192-198：RS、UPS、LMS + 6 * (MS + CS) 190、191 为备集群、193 为主集群 测试机：199（3MS）、201、202、203
测试目的	测试 OB 能否正常进行故障恢复		
测试输入	 数据表 schema： drop table if exists t1; create table t1 (c1 int primary key, c2 int, c3 varchar(50)); 插入 120w 条数据到系统中 故障恢复测试： 1. 杀掉主 ups，然后再重启，观察集群是否正常可服务 2. 杀掉所有 server，然后再全部重启并设主（设置日志最靠前的		

	ups 为主，两种方法确定主 ups：1.看日志文件多少；2.看 ups log，搜 commit point），观察集群是否正常可服务

编号	4	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192-198：RS、UPS、LMS + 6 * (MS + CS) 190、191 为备集群、193 为主集群 测试机：199（3MS）、201、202、203
测试目的	测试 OB 能否正常进行每日合并		
测试输入	数据表 schema： drop table if exists t1; create table t1 (c1 int primary key, c2 int, c3 varchar(50)); 字符串的平均长度为 25 向 OB 中插入数据（replace），100 * 6 个线程（主集群每个 MS 连接 100 个线程） 共插入 120w 条数据 作每日合并操作（三次，每次合并前 replace 120w 条数据） 每日合并的相关配置操作： ALTER SYSTEM SET merge_delay_interval = 1 SERVER_TYPE = CHUNKSERVER; ALTER SYSTEM SET min_major_freeze_interval = 1 SERVER_TYPE = UPDATESERVER; ALTER SYSTEM SET min_merge_interval = 1 SERVER_TYPE = UPDATESERVER; 每日合并的相关命令： ./rs_admin -r 10.11.1.193 -p 31500 get_obi_role ./ups_admin -a 10.11.1.193 -p 31701 -t major_freeze ./rs_admin -r 10.11.1.193 -p 31500 stat -o merge 进行多次换主（在高压下换主），每次换主后再进行每日合并（并且前一次每日合并还没有完成）。		

编号	5	配置	190-199：RS、UPS、MS、CS 测试机：199（3MS）、201、202、203 非三集群
测试目的	非 3 集群是否可以正常服务		
测试输入	 配置一个 2 集群，观察系统是否可以正常服务，性能是否正常 进行集群换主、故障恢复和每日合并操作。		

	数据表 schema： drop table if exists t1; create table t1 (c1 int primary key, c2 int, c3 varchar(50)); 每日合并的相关配置操作： ALTER SYSTEM SET merge_delay_interval = 1 SERVER_TYPE = CHUNKSERVER; ALTER SYSTEM SET min_major_freeze_interval = 1 SERVER_TYPE = UPDATESERVER; ALTER SYSTEM SET min_merge_interval = 1 SERVER_TYPE = UPDATESERVER; 每日合并的相关命令： ./rs_admin -r 10.11.1.190 -p 31500 get_obi_role ./ups_admin -a 10.11.1.190 -p 31701 -t major_freeze ./rs_admin -r 10.11.1.190 -p 31500 stat -o merge 查看集群角色和换主的相关命令： ./rs_admin -r 10.11.1.190 -p 31500 get_obi_role ./rs_admin -r 10.11.1.190 -p 31500 reelect 杀掉主集群 ups，再重启。 杀掉两个 ups 后，再重启。 配置一个 5 集群，重复上述测试，观察集群是否正常提供服务。 配置一个 7 集群，重复上述测试，观察集群是否正常提供服务。		
编号	6	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192-198：RS、UPS、LMS + 6 * (MS + CS) 190、191 为备集群、193 为主集群
测试目的	在换主的过程中建表，观察系统是否正常提供服务		

测试输入	数据表 schema：
	drop table if exists reelecttest; create table reelecttest(id int, value varchar(20), primary key(id));
	周期性换主，并在换主的过程压有少量的负载（几百条更新操作）
	
	在换主的过程中建表：
	create table t1(id int, value varchar(20), primary key(id));
	create table t2(id int, value varchar(20), primary key(id));		

	create table t3(id int, value varchar(20), primary key(id));
	create table t4(id int, value varchar(20), primary key(id));
	…
	
	select * from t1;
	select * from t2;
	select * from t3;
	select * from t4;
	…

可扩展提交

编号	1	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192-198：RS、UPS、LMS + 6 * (MS + CS) 190、191 为备集群、193 为主集群 测试机：199（3MS）、201、202、203
测试目的	测试在多集群日志强同步下，ScalableCommit 能否正常工作和其基本性能		
测试输入	向 OB 中插入数据（replace），100 * 6 个线程（主集群每个 MS 连接 100 个线程） 共插入 300W 条数据 数据表 Schema： drop table if exists t1; create table t1 (c1 int primary key, c2 int, c3 varchar(50)); 字符串的平均长度为 25		
编号	2	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192-198：RS、UPS、LMS + 6 * (MS + CS) 190、191 为备集群、193 为主集群 测试机：199（3MS）、201、202、203
测试目的	在多集群换主情况下，能够保证正确性和持续可服务		
测试输入	向 OB 中插入数据（replace），100 * 6 个线程（主集群每个 MS 连接 100 个线程） 计划插入 300W 条数据 数据表 Schema： drop table if exists t1; create table t1 (c1 int primary key, c2 int, c3 varchar(50)); 字符串的平均长度为 25		

	在高负载下强制集群换主（仅一次）：
	./rs_admin -r 10.11.1.193 -p 31500 get_obi_role
	./rs_admin -r 10.11.1.193 -p 31500 reelect
	
	set @@session.ob_query_timeout=9000000000;
	select count(*) from t1;		

编号	3	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192-198：RS、UPS、LMS + 6 * (MS + CS) 190、191 为备集群、193 为主集群 测试机：199（3MS）、201、202、203
测试目的	高负载下执行每日合并系统能否正常可服务		

测试输入	向 OB 中插入数据（replace），100 * 6 个线程（主集群每个 MS 连接 100 个线程）
	计划插入 300W 条数据
	
	数据表 Schema：
	drop table if exists t1;
	create table t1 (c1 int primary key, c2 int, c3 varchar(50));
	字符串的平均长度为 25
	
	在高负载下强制每日合并（仅一次）：
	./ups_admin -p 31701 -t major_freeze
	./rs_admin -r 10.11.1.193 -p 31500 stat -o merge
	
	使每日合并立刻执行
	ALTER SYSTEM SET merge_delay_interval = 1 SERVER_TYPE = CHUNKSERVER;
	ALTER SYSTEM SET min_major_freeze_interval = 1 SERVER_TYPE = UPDATESERVER;
	ALTER SYSTEM SET min_merge_interval = 1 SERVER_TYPE = UPDATESERVER;
	
	set @@session.ob_query_timeout=9000000000;
	select count(*) from t1;		

编号	4	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192-198：RS、UPS、LMS + 6 * (MS + CS) 190、191 为备集群、193 为主集群 测试机：199（3MS）、201、202、203
测试目的	测试系统能否正常进行故障恢复		
测试输入	向 OB 中插入数据（replace），100 * 6 个线程（主集群每个 MS 连接 100 个线程）		

	计划插入 300W 条数据 数据表 Schema： drop table if exists t1; create table t1 (c1 int primary key, c2 int, c3 varchar(50)); 字符串的平均长度为 25 待数据完全插入以后，在没有的负载的情况下进行每日合并： ALTER SYSTEM SET merge_delay_interval = 0 SERVER_TYPE = CHUNKSERVER; ./ups_admin -p 31701 -t major_freeze ./rs_admin -r 10.11.1.193 -p 31500 stat -o merge 确认每日合并完成后，杀掉所有系统进程（模拟机房断电），然后再重启所有进程并设主（还是设原来的主 UPS 为主）。		
编号	5	配置	190：RS、UPS、MS、CS 191：RS、UPS、MS、CS 192-198：RS、UPS、LMS + 6 * (MS + CS) 190、191 为备集群、193 为主集群 测试机：199（3MS）、201、202、203
测试目的	在高负载下，测试 DDL 操作的影响		

测试输入	向 OB 中插入数据（replace），100 * 6 个线程（主集群每个 MS 连接 100 个线程）
	计划插入 300W 条数据
	
	数据表 Schema：
	drop table if exists t1;
	create table t1 (c1 int primary key, c2 int, c3 varchar(50));
	字符串的平均长度为 25
	
	在高负载执行如下建表删表操作：
	create table t2 (c1 int primary key, c2 int, c3 varchar(50));
	create table t3 (c1 int primary key, c2 int, c3 varchar(50));
	drop table t2;
	show tables;		