慢sql查询case

case1:

【P0】RPC:DemanderInitNewbieTask Unexpected Exception

【环境信息】：(PSM: ad.star.marketing, ENV: prod, USER: , TCE_HOST_ENV: online, IDC: LF_YJD, POD_IP: 10.13.236.173)

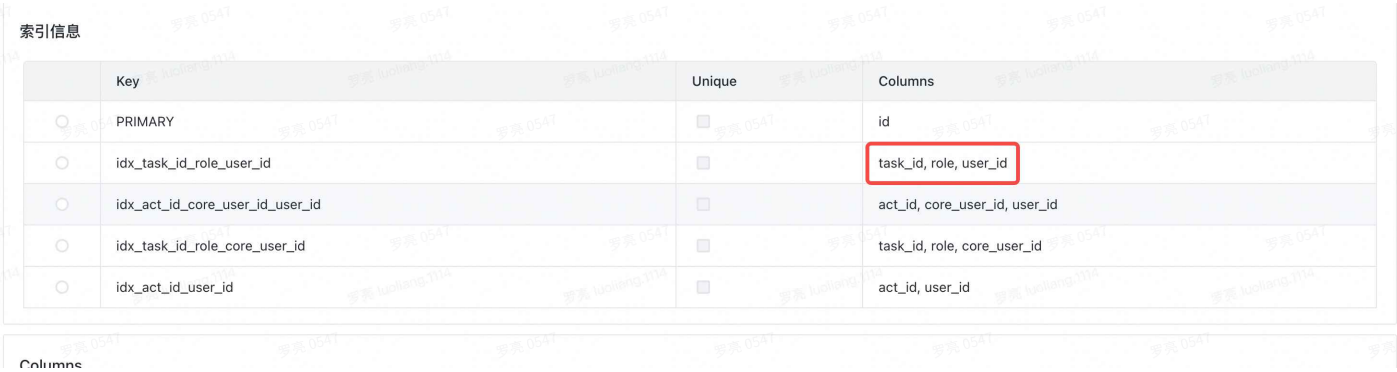
【报警内容】：
异常服务：DemanderInitNewbieTask，异常信息：Internal system error. Please contact the operator，请求参数：DemanderInitNewbieTaskReq(Base=BASEBase(Addr=u'10.112.0.139', Extra={u'tracestate': u'_sr=1.0', u'cluster': u'default', u'env': u'prod', u'traceparent': u'02-170895b3944d538b314fd5e0a3c68104-76db66c81d224b5c-01', u'byted-trace-id': u'170895b3944d538b314fd5e0a3c68104:76db66c81d224b5c:3aa7135ad7b352d4:1', u'jaeger-baggage': u'_sr=1.0'}, Caller=u'ad.star.http', LogID=u'2022071322351501020803809015137 3D5', TrafficEnv=None, Client=u''), s_demander_id=1719762726222861)

【trace id】：2022071322351501020803809015137 3D5

【错误追踪】：
Traceback (most recent call last):
File "/opt/tiger/toutiao/app/ad/star_common/utils/auto_handler.py", line 137, in _wrap_rpc
resp = star_rpc_post(handler)(**params) or {}
File "/opt/tiger/toutiao/app/ad/star_common/decorators/rpc.py", line 170, in wrap
return supplier_db_commit(star_data_db_commit(func))(*args, **kwargs) or {}
File "/opt/tiger/toutiao/app/ad/star_core/db/session.py", line 301, in wrapper
raise e
OperationalError: (OperationalError) (1105, 'EOF') 'SELECT star_act_user_task.create_time AS star_act_user_task_create_time, star_act_user_task.modify_time AS star_act_user_task_modify_time, star_act_user_task.id AS star_act_user_task_id, star_act_user_task.act_id AS star_act_user_task_act_id, star_act_user_task.task_id AS star_act_user_task_task_id, star_act_user_task.role AS star_act_user_task_role, star_act_user_task.user_id AS star_act_user_task_user_id, star_act_user_task.core_user_id AS star_act_user_task_core_user_id, star_act_user_task.status AS star_act_user_task_status, star_act_user_task.start_time AS star_act_user_task_start_time, star_act_user_task.end_time AS star_act_user_task_end_time, star_act_user_task.`rank` AS `star_act_user_task_rank`, star_act_user_task.`visible` AS `star_act_user_task_visible`, star_act_user_task.data AS star_act_user_task_data, star_act_user_task.deleted AS star_act_user_task_deleted \nFROM star_act_user_task \nWHERE star_act_user_task.user_id = %s AND star_act_user_task.task_id =

%s AND star_act_user_task.deleted = %s ORDER BY star_act_user_task.id DESC \n LIMIT %s' (1719762726222861, 7111953664180224030L, 0, 1)

表索引情况：


索引信息

	Key	Unique	Columns
○	PRIMARY	□	id
○	idx_task_id_role_user_id	□	task_id, role, user_id
○	idx_act_id_core_user_id_user_id	□	act_id, core_user_id, user_id
○	idx_task_id_role_core_user_id	□	task_id, role, core_user_id
○	idx_act_id_user_id	□	act_id, user_id

Columns

原因： order_by和where给优化器提供了两个索引选择，优化器选择走了主键索引，而没走联合索引

导致：

	A	B
1	id	1
2	select_type	SIMPLE
3	table	star_act_user_task
4	partitions	
5	type	index
6	possible_keys	idx_task_id_role_user_id,idx_task_id_role_core_user_id
7	key	PRIMARY
8	key_len	8
9	ref	
10	rows	246
11	filtered	0.02
12	Extra	Using where

解决办法：

- 如果是兜底策略：去掉order_by
- 构建联合索引 user_id+task_id+deleted+id

---

case2:

/* psm=ad.star.challenge, ip=10.152.49.31, ip=- */ SELECT /* psm=ad.star.challenge, ip=unknown, pid=228 */ item_cost FROM `star_item_cost_daily` WHERE (challenge_id = 7117184360545468423) AND (deleted = 0) ORDER BY item_cost desc LIMIT 1

	A	B
1	id	1
2	select_type	SIMPLE
3	table	star_item_cost_daily
4	partitions	
5	type	ref
6	possible_keys	idx_challenge_item
7	key	idx_challenge_item
8	key_len	8
9	ref	const
10	rows	66120
11	filtered	10
12	Extra	Using index condition; Using where; Using filesort

	A	B	C
1	idx_challenge_item		challenge_id, item_id

使用challenge_id, deleted, item_cost这样的组合索引是不是更好

SELECT item_cost FROM `star_item_cost_daily` WHERE (challenge_id = 7117184360545468423) AND (deleted = 0) ORDER BY item_cost desc LIMIT 1，这条成为慢sql的原因是order by item_cost，触发了排序。如果同一个challenge_id下有很多行的话，估计排序还不能在内存中进行，所以耗时就长了。加了组合索引后有几个好处，1. 不需要排序，原先扫描条数估计是对6w行排序后返回第一行，现在只需要扫两条直接返回就行；2. 不需要回表，原来的deleted必须回表后取出来由mysql服务层判断

两个点注意：

- deleted字段基本属于必会使用的where，如果能加入联合索引，可以减少回表的操作，回表的消耗主要体现在磁盘io，需要把磁盘页调入内存
- 索引数量问题，索引太多的坏处就是，优化器可能会做错误的选择，其次优化器基于成本的分析的耗时可能也会延长

case3:

```
/* psm=ad.star.data, ip=10.112.22.161, ip=- */ SELECT /* psm=ad.star.data,ip=10.11
FROM star_author_manage_demander_industry
WHERE star_author_manage_demander_industry.sync_date >= '2022-08-15' AND star_auth
LIMIT 1;
```

分析详情（时间单位为 s）

Rank	指纹	执行次数	平均耗时	最大耗时	来源	执行实例	查看详情
1	select count(star_author_manage_demander_industry.id) as count_id from star_author_manage_demander_industry whe...	13	4.048530917901259	6.013525	ad.star.data	10.224.188.29:3306	查看详情

	A	B
1	id	1
2	select_type	SIMPLE
3	table	star_author_manage_demander_industry
4	partitions	
5	type	range
6	possible_keys	idx_sync_date
7	key	idx_sync_date
8	key_len	4
9	ref	
10	rows	41127
11	filtered	10
12	Extra	Using index condition; Using where

慢查询原因：

- 由于索引只包含sync_date字段，查询条件还需对deleted进行判断，所以每次查询都需要回表。
- 且由于二级索引会打乱主见索引原始顺序，回表过程是一个随机io都过程，会造成频繁的缺页和刷盘
- 涉及行数41127行，需回表量大

解决方案

- 方案1：由于该表是hive2mysql写入，不存在deleted=1的情况，可以从sql语句出发，删除deleted判断条件
- 方案2：利用索引下推机制，构建deleted和sync_date的联合索引