MySQL Automatic Diagnostic System, Mechanism and Usage

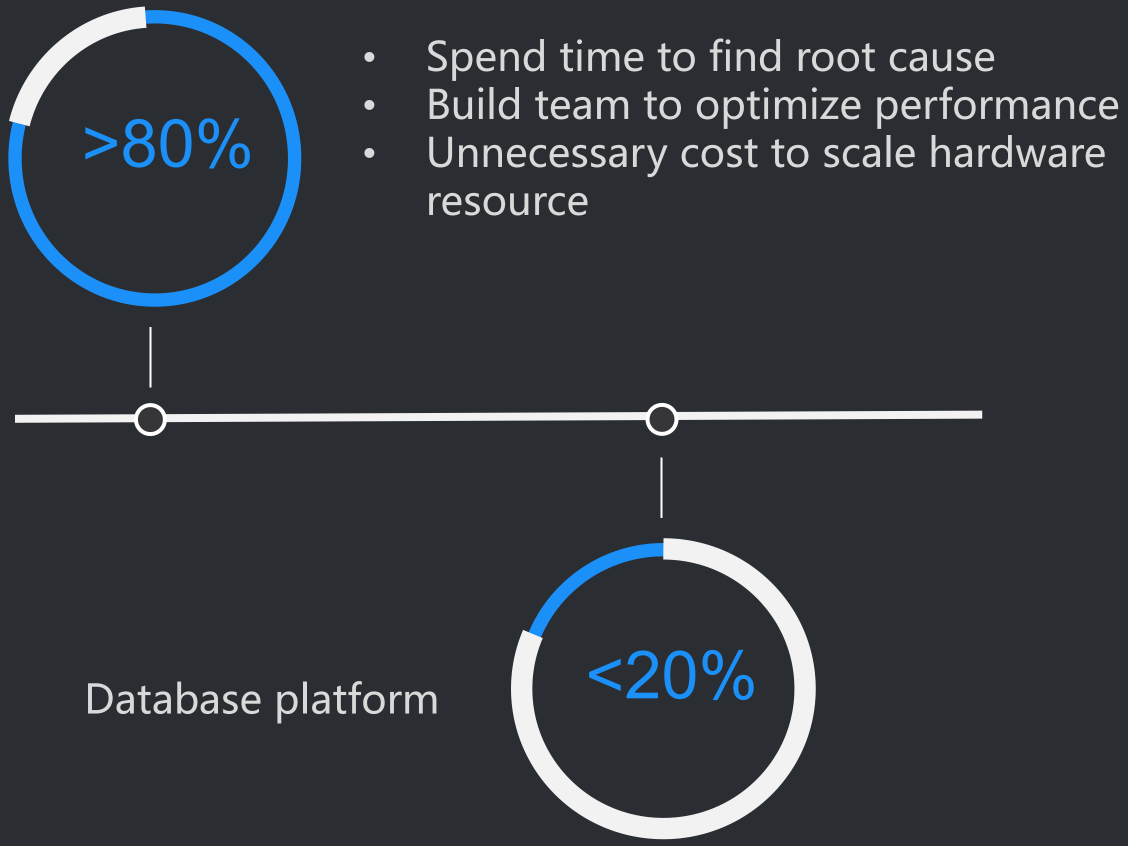
Shangshun Lei, Alibaba Cloud
Lixun Peng, Alibaba Cloud

Agenda

- Why CloudDBA
- Architecture
- Online Diagnosis
- Offline Diagnosis
- SQL Advisor

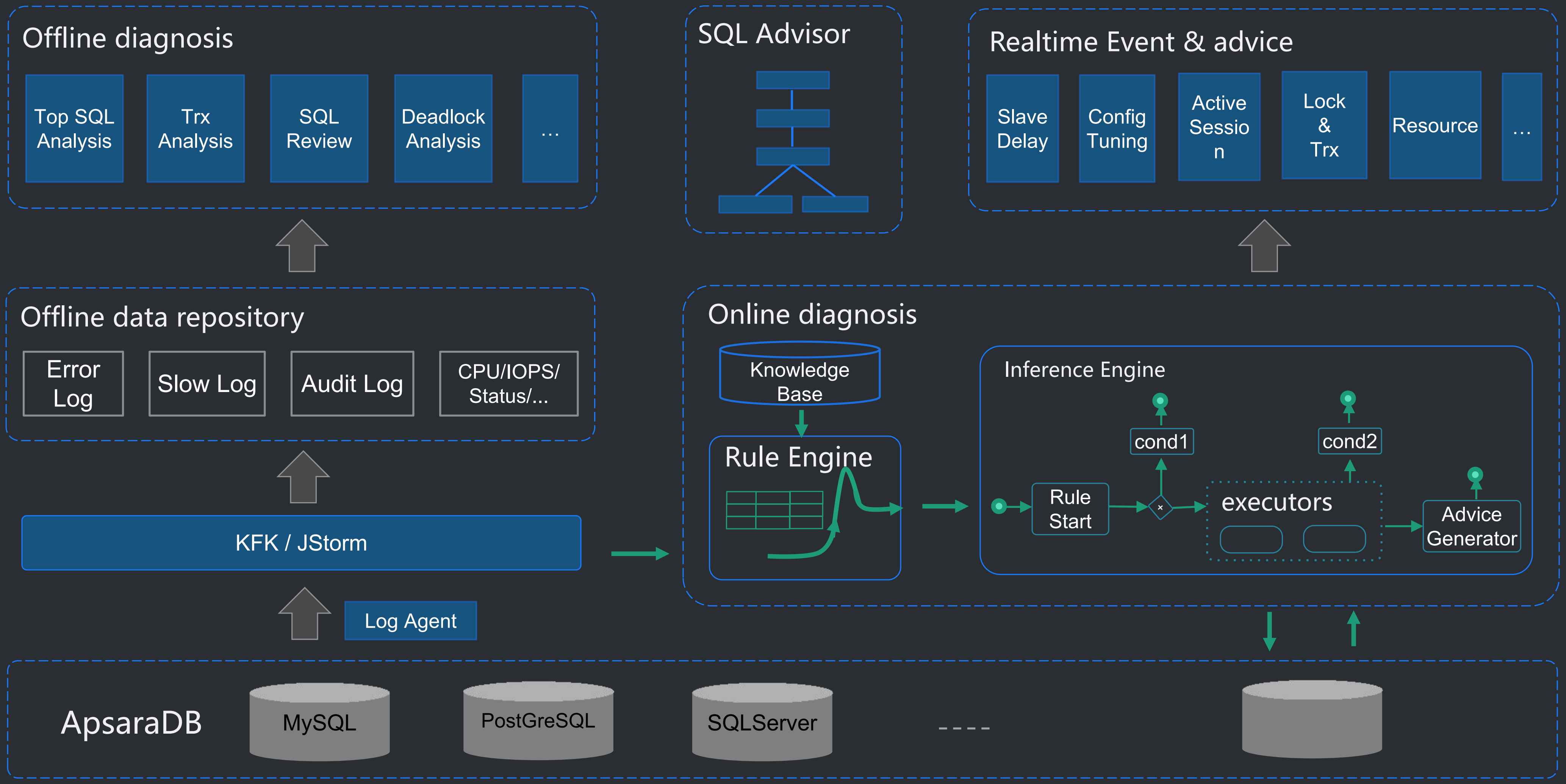
Why CloudDBA ?

- Reduce cost and we do care about it
- Focus your resources on business
- Provide best technology



- Spend time to find root cause
- Build team to optimize performance
- Unnecessary cost to scale hardware resource

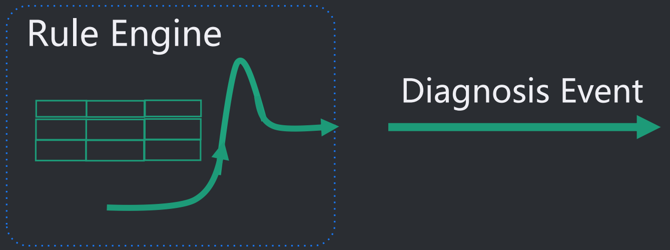
CloudDBA Architecture



Online diagnosis

- Rule Engine

1. Immediate detection of useful changes with low cost
2. Choose correct inference model

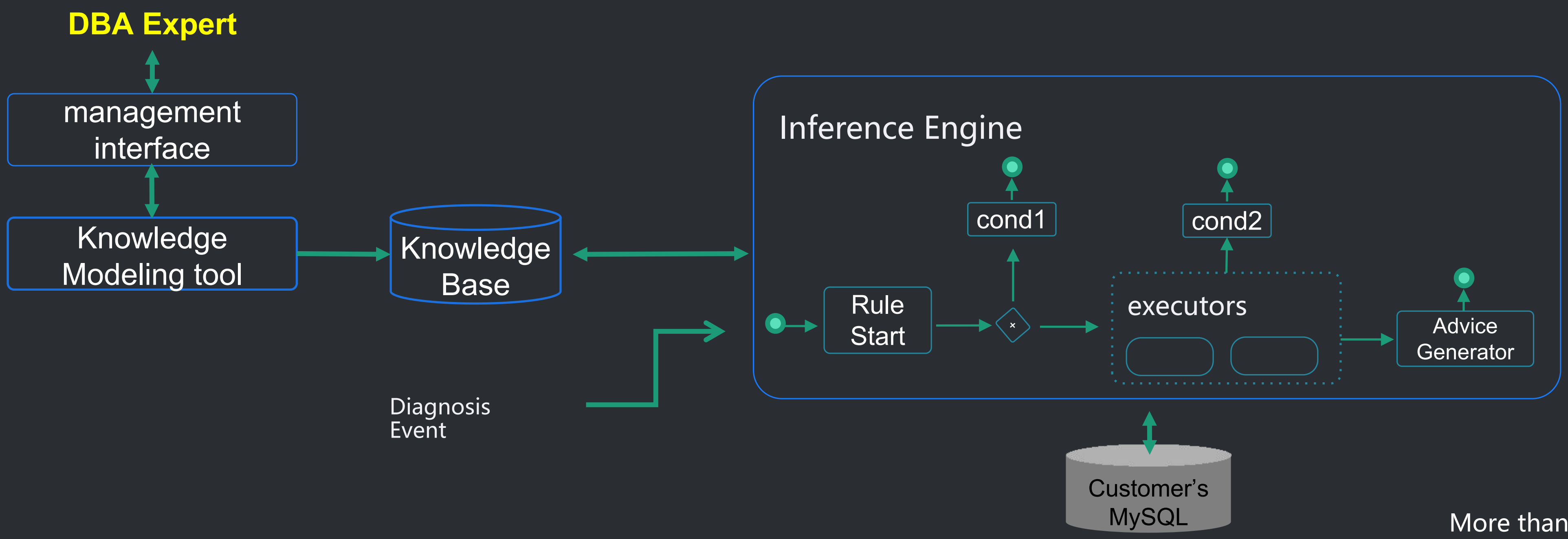


1. Database global status is maturity and easy to get
2. High frequency monitoring to make sure no useful info missed
3. Real time state change detection algorithms
4. Importance of database experience

Online diagnosis

- Knowledge Base & inference Engine

1. Ability to accumulate DBA experts' experience in short time
2. Accurate issue detection & corresponding advice



Offline diagnosis

- Audit log does matter

1. Record full SQLs for database

1. A feature of AliSQL, no performance impact

1. Can only be used with customer’ s authorization

Offline diagnosis

- Transaction analysis

1. Uncommitted transactions

1. Long transactions

1. Long interval between transaction SQLs

4. Big transactions

Offline diagnosis

- SQL review

1. How many types of SQLs
2. How many types of Transactions
3. SQLs or sequence in transaction is expected or not
4. Scan rows, return rows, elapsed time & SQL advice

Offline diagnosis

- Top SQLs

  1. Need to get top SQLs before optimize

  2. Help to explain questions such as "why my CPU is 100%

  3. Different statistics dimensions & performance metrics

SQL Advisor

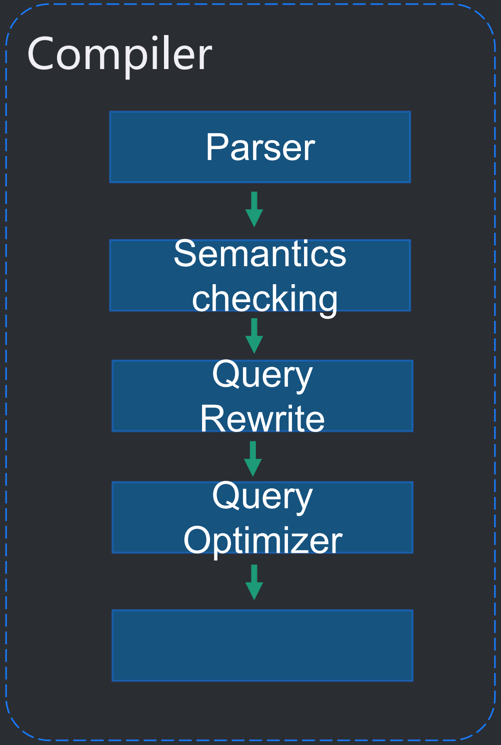
- Not kernel built in component, externally implemented
- Not database optimizer, but help optimizer to find the best execution path

Query Rewriter

— What to do

Query Optimizer

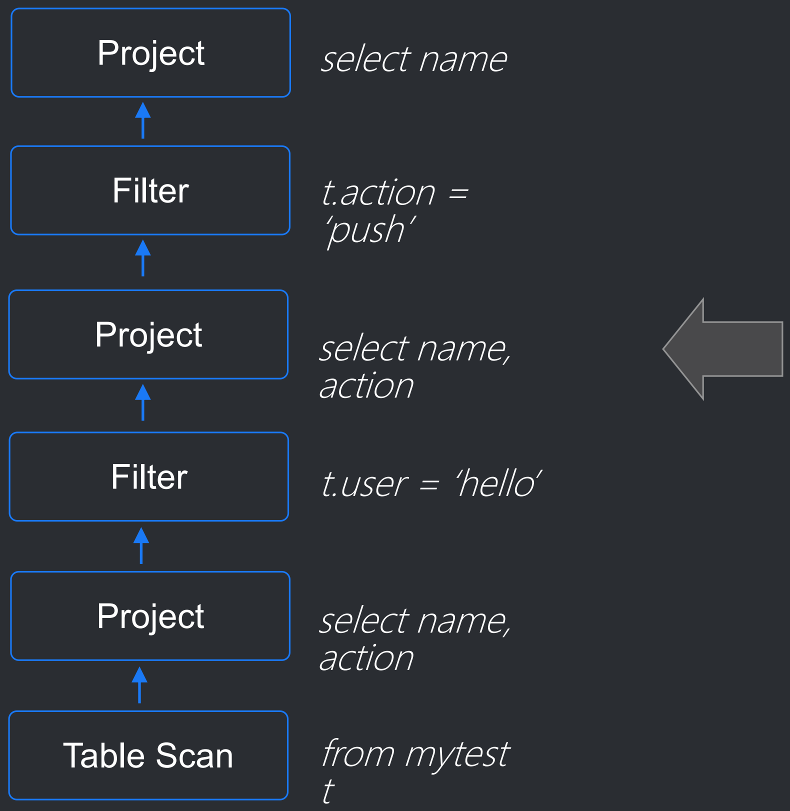
— How to do

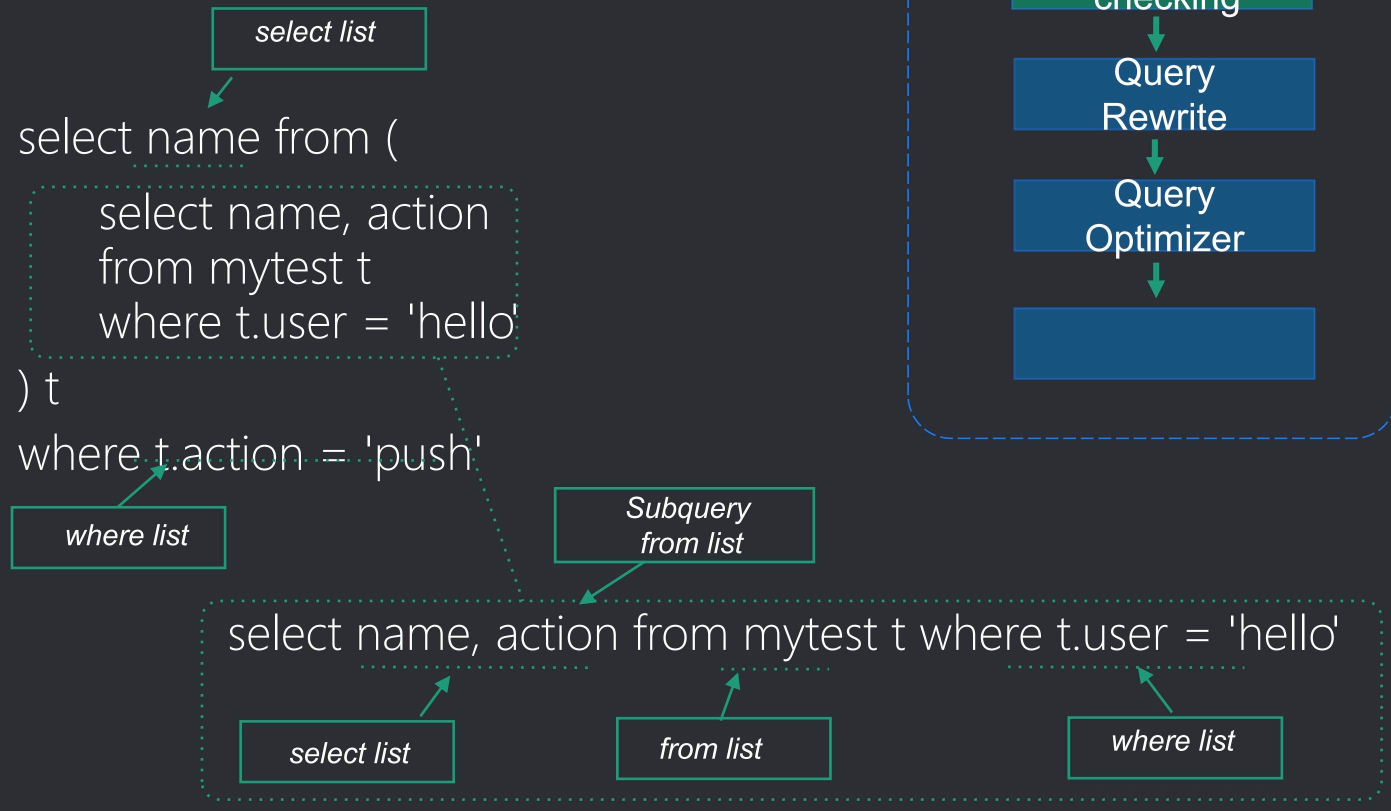


SQL Advisor

How to get SQL optimized ?

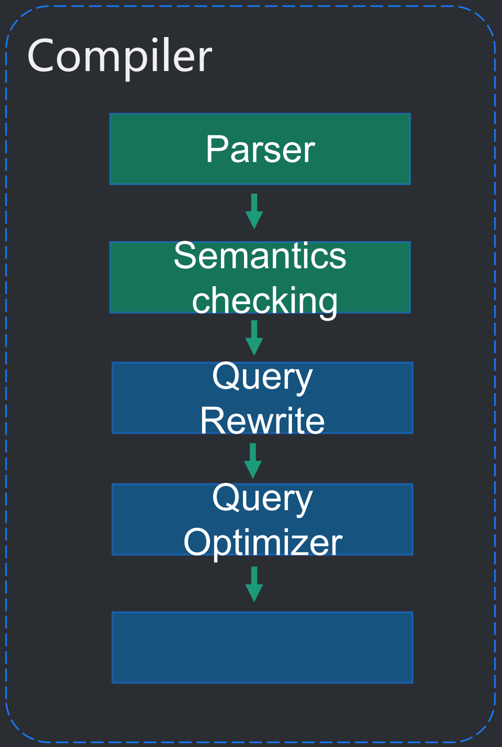
- Different view for SQL





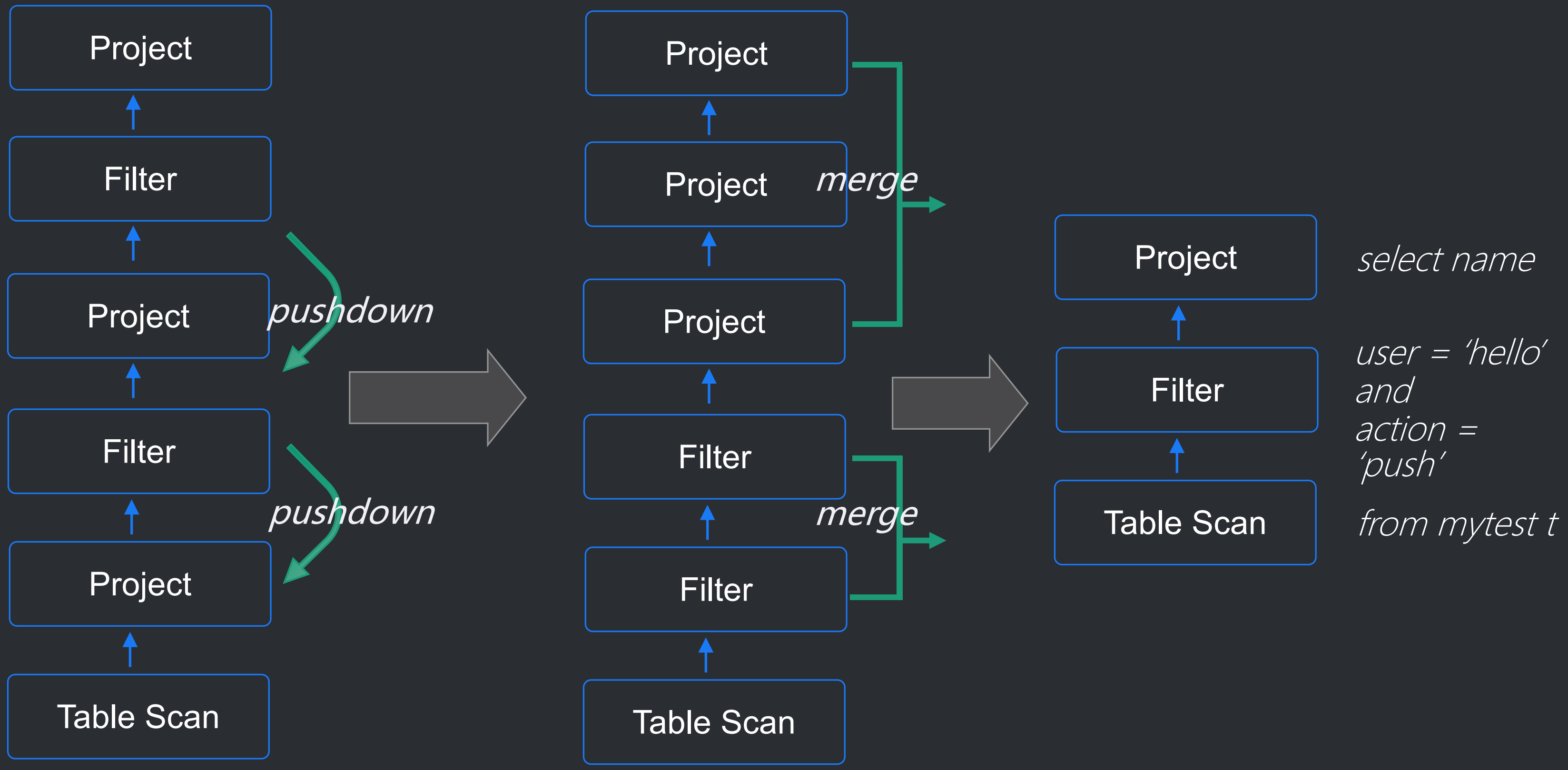
```
select name from (
    select name, action
    from mytest t
    where t.user = 'hello'
) t
where t.action = 'push'
```

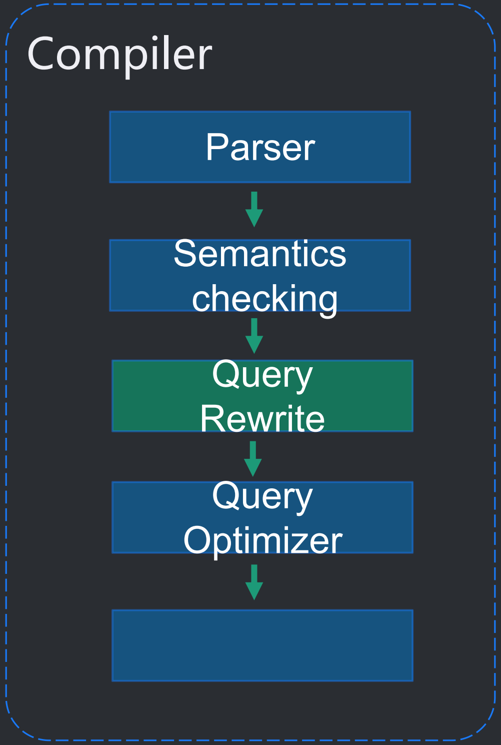
```
select name, action from mytest t where t.user = 'hello'
```



SQL Advisor

- Rewrite SQL with rules





SQL Advisor

- Follow rules to detect scenarios that index can not be applied

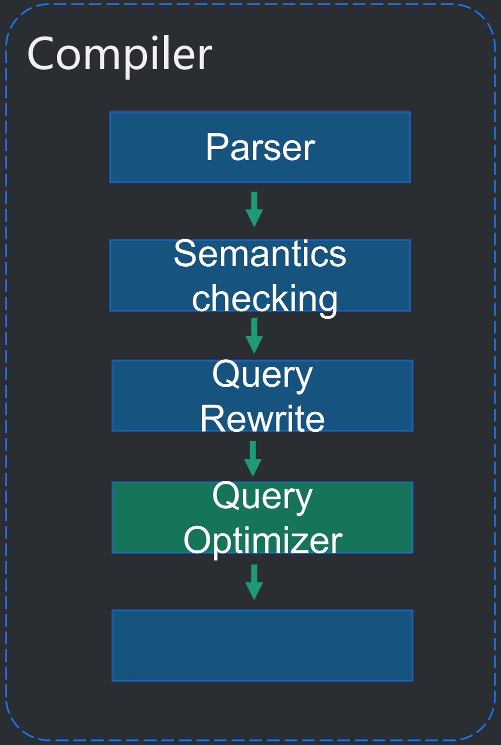
1. **Like expression with leading wildcards**

   first_name LIKE concat('%' , 'lei');
2. **Column as function argument**

   UPPER(first_name)
3. **Implicit conversion due to data type mismatch**

   a = 123
4. **Character set /collation mismatch**

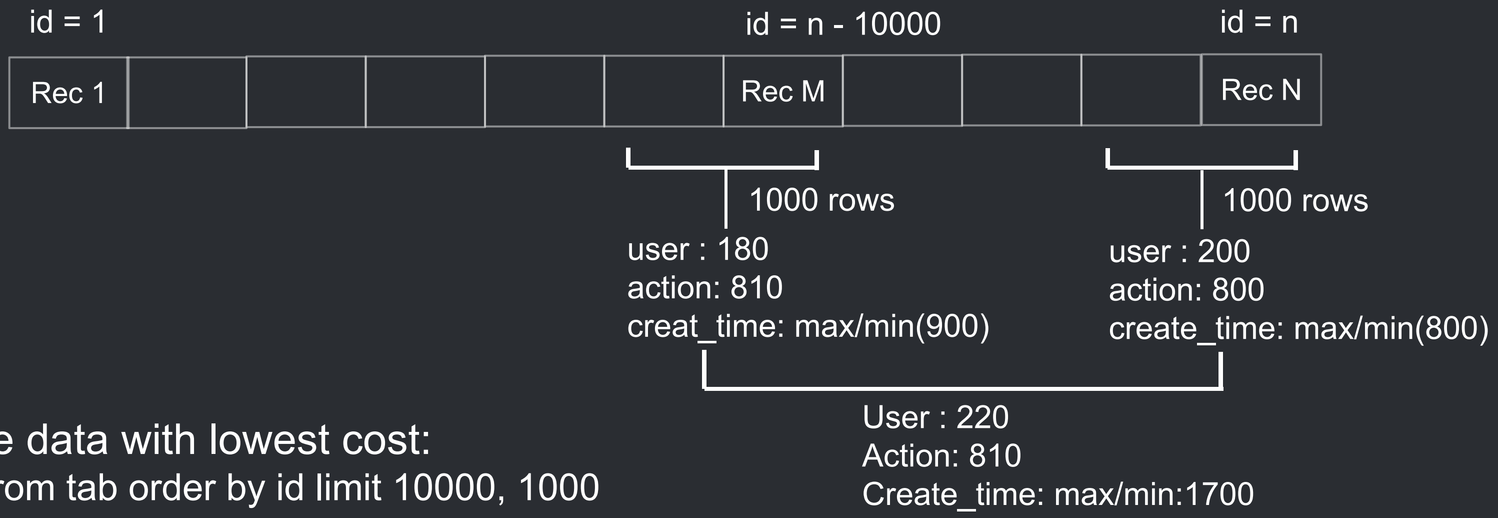
   t1.utf8_string = t2.utf8_bin_string

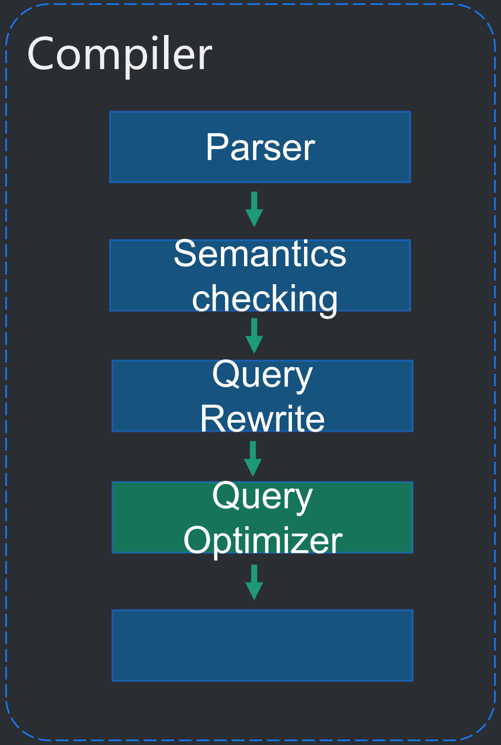


SQL Advisor

- Follow rules to create index with lowest cost

  1. Selectivity: estimated by sampling latest data





Get sample data with lowest cost:
  select * from tab order by id limit 10000, 1000

Distinct values:
  user or action: max distinct values
  create_time: (total records) * (avg distinct values) / 10000

Predicate selectivity:
  user, action: 1/ 220, 1/810
  create_time: 10000 / (n * 805)

SQL Advisor

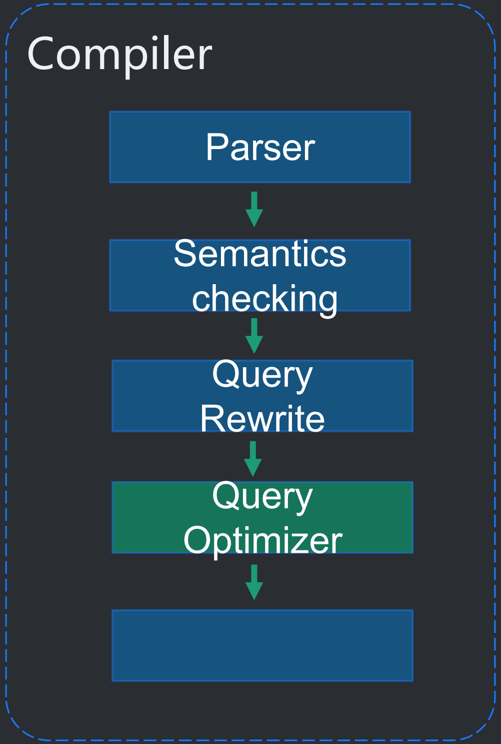
- Follow rules to create index with lowest cost

  1. Selectivity

     Conjunction: selectivity 1 * selectivity 2

     Disjunction: selectivity 1 + selectivity 2

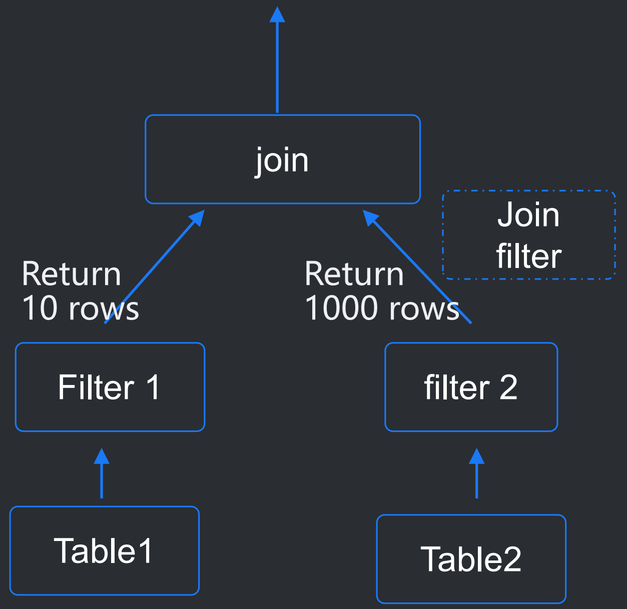
     ——Special handling for "LIMIT N"

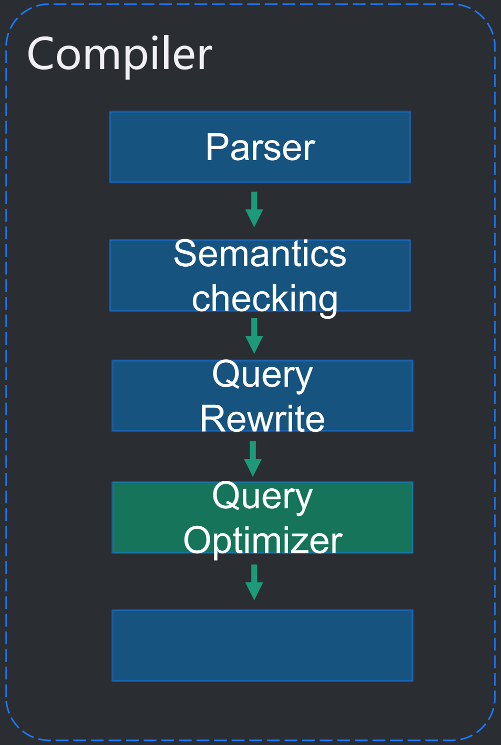


SQL Advisor

- Follow rules to create index with lowest cost

  2. Join method & order





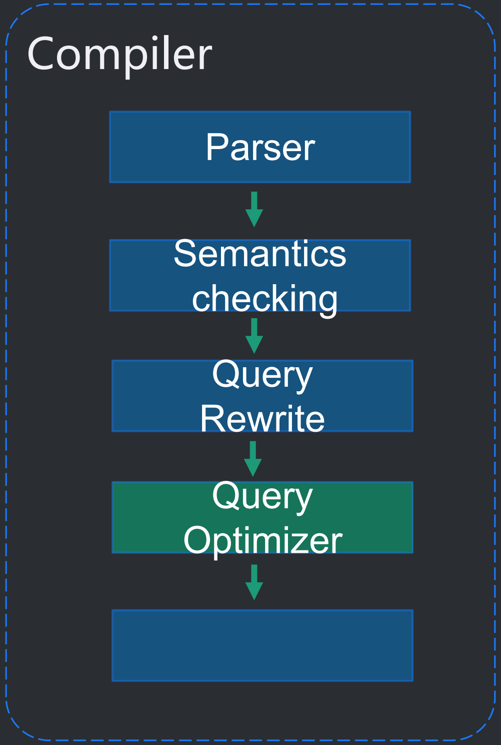
1. For nest loop, small table drive big table
2. Once drive table chosen, join filter replace filter2 for index

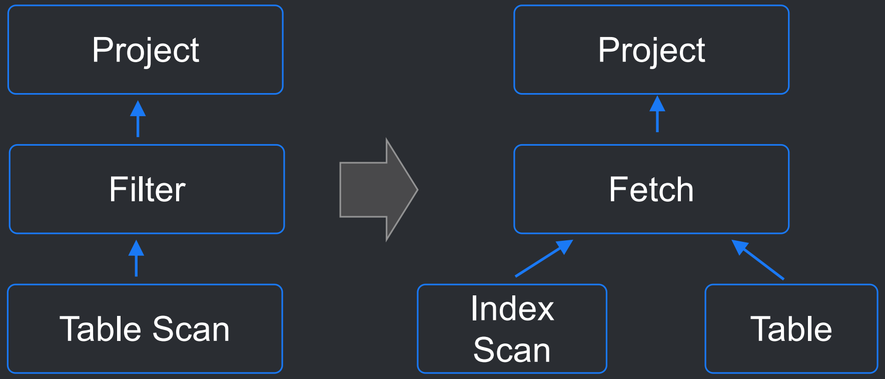
SQL Advisor

- Follow rules to create index with lowest cost

  3. Create index with predicates candidates

     - **Lower selectivity predicate first**
     - **Only 1 range predicate after equal**
     - **Consider use index for sort**
     - **Covered index**
     - **Limit index card**





Rewritten Advice:

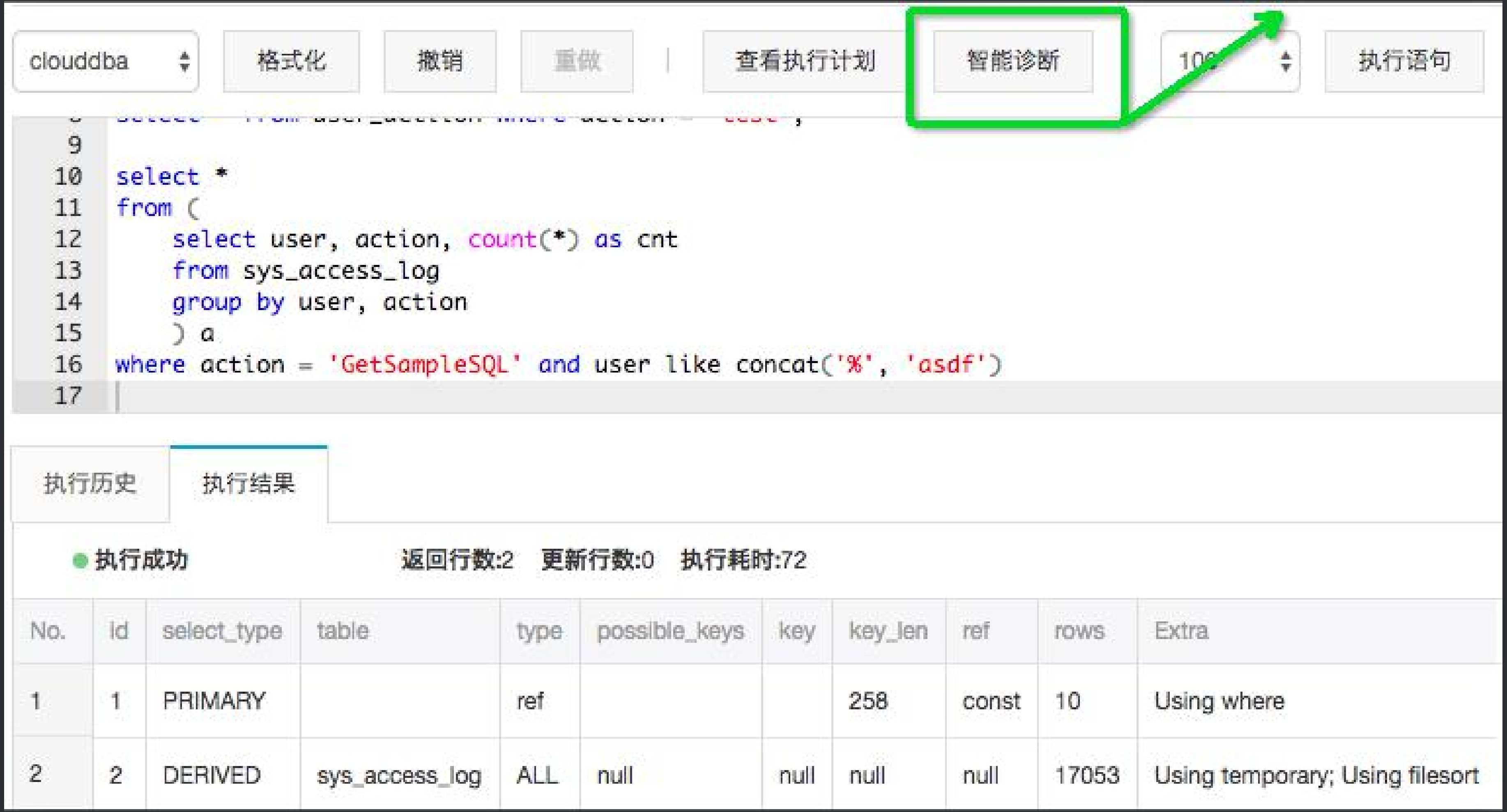
select name from mytest where user = 'hello' and action = 'push'

Index Advice:

ALTER TABLE `mytest` ADD INDEX rds_idx_1(user, action)

SQL Advisor

- Example



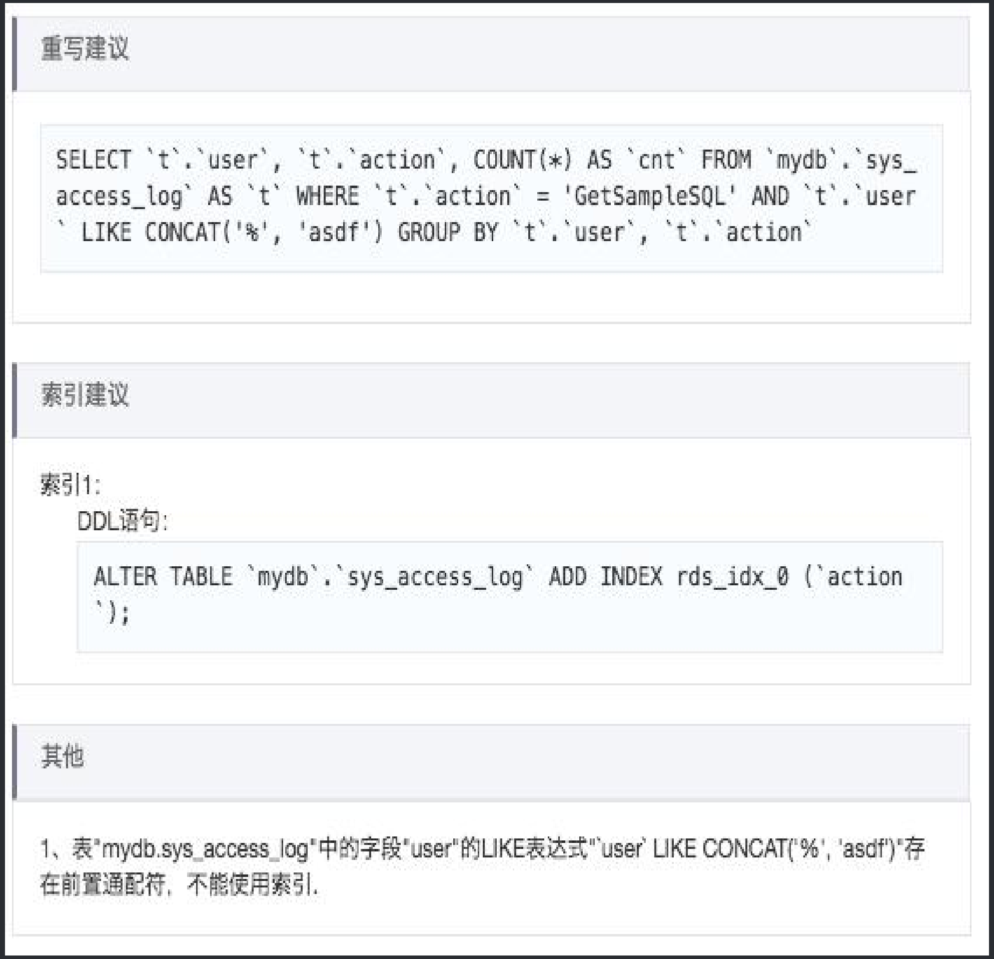
```
select *
from (
    select user, action, count(*) as cnt
    from sys_access_log
    group by user, action
    ) a
where action = 'GetSampleSQL' and user like concat('%', 'asdf')
```

● 执行成功　　返回行数:2　更新行数:0　执行耗时:72

No.	id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	1	PRIMARY		ref			258	const	10	Using where
2	2	DERIVED	sys_access_log	ALL	null	null	null	null	17053	Using temporary; Using filesort

SQL Advisor

- Example



重写建议

```
SELECT `t`.`user`, `t`.`action`, COUNT(*) AS `cnt` FROM `mydb`.`sys_
access_log` AS `t` WHERE `t`.`action` = 'GetSampleSQL' AND `t`.`user
` LIKE CONCAT('%', 'asdf') GROUP BY `t`.`user`, `t`.`action`
```

索引建议

索引1:
DDL语句:

```
ALTER TABLE `mydb`.`sys_access_log` ADD INDEX rds_idx_0 (`action
`);
```

其他

1、表"mydb.sys_access_log"中的字段"user"的LIKE表达式"`user` LIKE CONCAT('%', 'asdf')"存在前置通配符，不能使用索引.

Rewrite advice:

select `t`.`user`, `t`.`action`, count(*) as `cnt`
from `mydb`.`sys_access_log` `t`
where `t`.`action` = 'GetSampleSQL'
and `t`.`user` like concat('%', 'asdf')
group by `t`.`user`, `t`.`action`

Index advice:

ALTER TABLE `mydb`.`sys_access_log` ADD INDEX rds_idx_0 (`action`);

Other advice:

LiKE expression CONCAT('%', 'asdf') for column "user" of table "sys_access_log" with leading wildcard can not use index.

Thank you !

Q&A