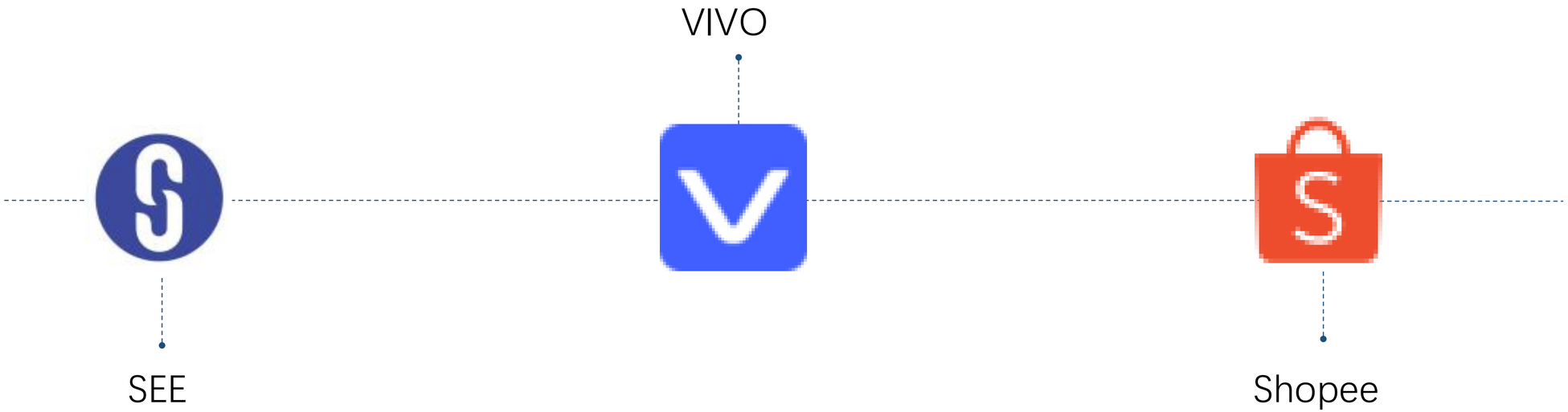
开源社
kaiyuanshe

SQL优化之美

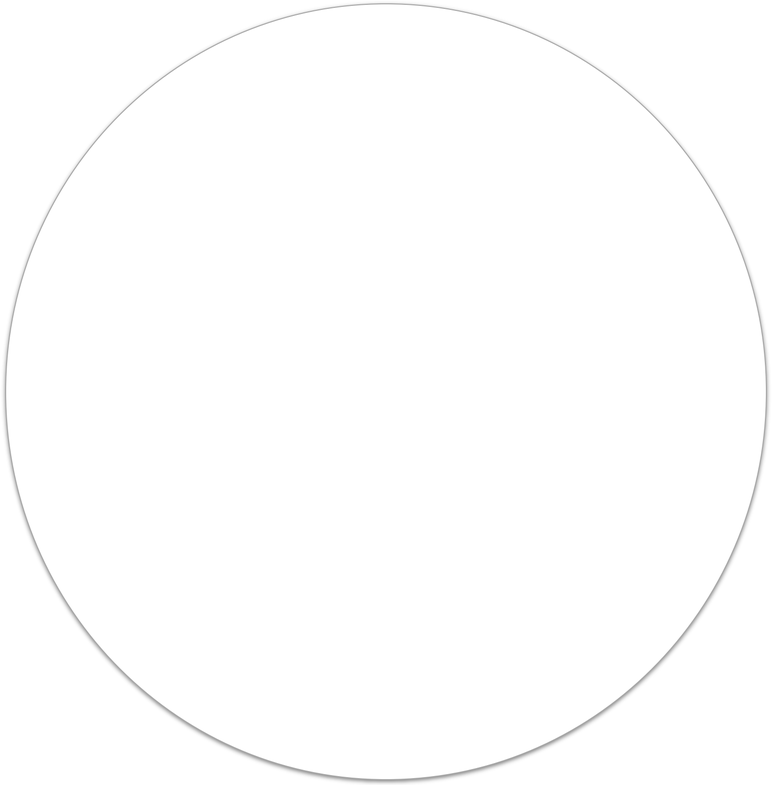
陈鹤

Shopee大数据专家


VIVO
SEE
Shopee

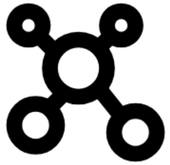
SQL——我们的老朋友

- 数据管理
- 数据存储
- 数据操作



- ETL
- 即席查询
- 后端开发

- 金融行业
- 社交媒体
- 电商



- MySQL
- Spark SQL
- Flink SQL

为什么是SQL?

标准化语言

- 跨平台兼容
- 极佳的移植性
- 减少学习成本

Reserved Words

The following keywords are also reserved words: Portable applications must not use them as user-defined names. An implementation may allow the use of any or all of these keywords, but flags such usage:

ABSOLUTE *	CROSS *	GLOBAL *	NOT	SOME
ACTION *	CURRENT	GO	NULL	SPACE *
ADD	CURRENT_DATE	GOTO	NULLIF *	SQL
ALL	CURRENT_TIME	GRANT	NUMERIC	SQLCA *
ALLOCATE	CURRENT_TIMESTAMP	GROUP	OCTET_LENGTH	SQLCODE *
ALTER	CURRENT_USER	HAVING	OF	SQLERROR
AND	CURSOR	HOUR	ON	SQLSTATE *
ANY	DATE	IDENTITY *	ONLY	SQLWARNING
ARE *	DAY	IMMEDIATE	OPEN	SUBSTRING
AS	DEALLOCATE	IN	OPTION	SUM
ASC	DEC	INCLUDE	OR	SYSTEM_USER
ASSERTION *	DECIMAL	INDEX	ORDER	TABLE
AT *	DECLARE	INDICATOR	OUTER	TEMPORARY *
AUTHORIZATION	DEFAULT	INITIALLY *	OUTPUT *	THEN *
AVG	DEFERRABLE *	INNER	OVERLAPS	TIME
BEGIN	DEFERRED *	INPUT	PAD *	TIMESTAMP
BETWEEN	DELETE	INSENSITIVE *	PARTIAL *	TIMEZONE_HOUR *
BIT *	DESC	INSERT	POSITION	TIMEZONE_MINUTE *
BIT_LENGTH *	DESCRIBE	INT	PRECISION	TO
BOTH	DESCRIPTOR	INTEGER	PREPARE	TRAILING
BY	DIAGNOSTICS	INTERSECT *	PRESERVE *	TRANSACTION
CASCADE	DISCONNECT	INTERVAL	PRIMARY	TRANSLATE
CASCADED *	DISTINCT	INTO	PRIOR *	TRANSLATION *
CASE *	DOMAIN *	IS	PRIVILEGES	TRIM
CAST	DOUBLE	ISOLATION	PROCEDURE *	TRUE *
CATALOG *	DROP	JOIN	PUBLIC	UNCOMMITTED
CHAR	ELSE *	KEY	READ	UNION
CHARACTER	END	LANGUAGE *	REAL	UNIQUE
CHARACTER_LENGTH	END-EXEC *	LAST *	REFERENCES	UNKNOWN *
CHAR_LENGTH	ESCAPE	LEADING	RELATIVE *	UPDATE
CHECK	EXCEPT *	LEFT	REPEATABLE	UPPER
CLOSE	EXCEPTION	LEVEL	RESTRICT	USAGE *
COALESCE *	EXEC	LIKE	REVOKE	USER
COLLATE	EXECUTE	LOCAL *	RIGHT	USING
COLLATION	EXISTS	LOWER	ROLLBACK	VALUE
COLUMN	EXTERNAL *	MATCH *	ROWS *	VALUES
COMMIT	EXTRACT	MAX	SCHEMA	VARCHAR
COMMITTED	FALSE *	MIN	SCROLL *	VARYING
CONNECT	FETCH	MINUTE	SECOND	VIEW
CONNECTION	FIRST *	MODULE *	SECTION	WHEN *
CONSTRAINT *	FLOAT	MONTH	SELECT	WHENEVER
CONSTRAINTS *	FOR	NAMES *	SERIALIZABLE	WHERE
CONTINUE	FOREIGN	NATIONAL	SESSION *	WITH
CONVERT *	FOUND	NATURAL *	SESSION_USER	WORK
CORRESPONDING *	FROM	NCHAR	SET	WRITE *
COUNT	FULL *	NEXT *	SIZE *	YEAR
CREATE	GET	NO *	SMALLINT	ZONE *

* X/Open SQL does not use these keywords, but they are defined in the International Standard.

声明式编程

命令式编程

- 描述“如何做”
- 受语言、框架、版本的限制

```
class User(val phoneNumber: String)

// ...
def findUsersWithPhoneNumberStartingWith(userList: Array[User], prefix: String): Array[User] = {
    userList.filter(user => user.phoneNumber.startsWith(prefix))
}
```

查找手机号155开头的用户

声明式编程

- 描述“做什么”
- 支持多种底层实现策略

```
SELECT *
FROM users
WHERE phone LIKE '155%';
```

成也SQL，败也SQL

我们以为的SQL

声明式

自优化

学习成本极低

简直无所不能

实际的SQL

SQL跑不动

无法用SQL实现

复杂逻辑和祖传代码

表达能力有限

直播间观看时长

- 进入直播间后开始上报埋点
- 5秒一次心跳
- 没有退出事件，仅能依靠session

连续登录

- 一周内累计登录时长超过一小时
- 剔除时长小于10秒的用户

加购、下单数

- 动线为浏览商品、加购、下单
- 有多少用户在中间步骤放弃购买
- 操作时效24小时

信用卡消费

- 最近三个月内最长连续消费的天数

易学难精

在order_type有倾斜的情况下哪种效率更高?

```
SELECT SPLIT(first_phase_type, '-')[1] AS second_phase_type
            ,SUM(cnt)
            ,SUM(amt)
FROM (SELECT first_phase_type
                    ,COUNT(1) AS cnt
                    ,SUM(amount) AS amt
            FROM (SELECT concat(CAST(RAND() * 90 + 10 AS INT), '-' , order_type)  AS first_phase_type
                                ,amount
                        FROM `order`) t1
            GROUP BY first_phase_type) t2
GROUP BY SPLIT(first_phase_type, '-')[1];
```

```
SELECT order_type
            ,COUNT(1)
           ,SUM(amount)
FROM `order`
GROUP BY order_type;
```

为什么要调优?

- 降本增效
  - 降低资源开销
  - 提升开发效率
- 自我提升
- 不能"只得其形，未得其意”

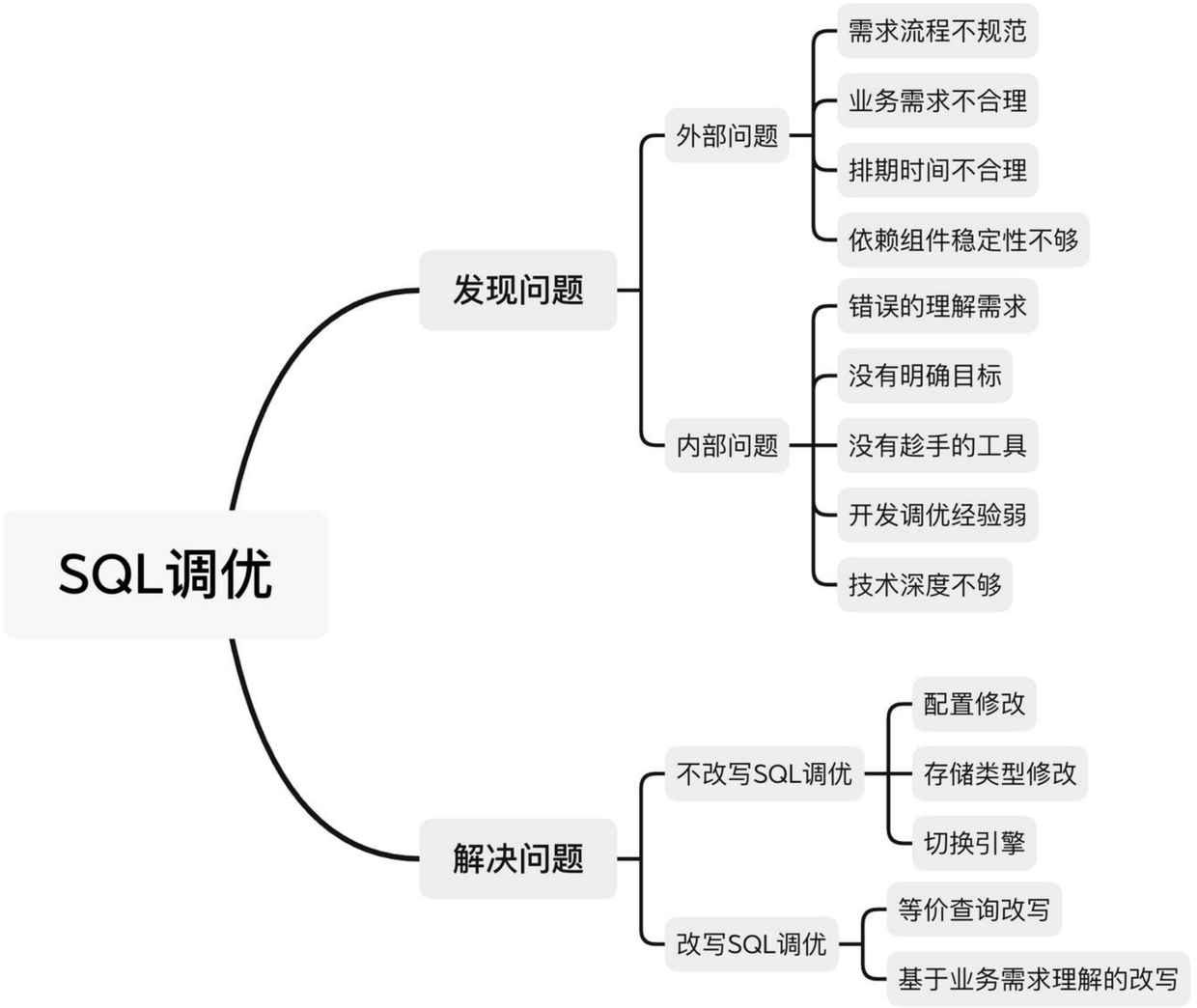
降本增效

一个索引，报表加载时间提升28.7倍

Name	Status	Type	Initiator	Size	Time
execute	200	fetch	vendors.005932c....js:2	4.6 kB	12.43 s
execute	200	fetch	vendors.005932c....js:2	2.8 kB	12.42 s
execute	200	fetch	vendors.005932c....js:2	7.0 kB	13.80 s
execute	200	fetch	vendors.005932c....js:2	706 B	9.71 s
execute	200	fetch	vendors.005932c....js:2	717 B	13.32 s
execute	200	fetch	vendors.005932c....js:2	718 B	12.16 s
json	200	fetch	vendors.005932c....js:2	209 B	54 ms
execute	200	fetch	vendors.005932c....js:2	716 B	5.16 s
json	200	fetch	vendors.005932c....js:2	209 B	65 ms
execute	200	fetch	vendors.005932c....js:2	9.3 kB	8.91 s
execute	200	fetch	vendors.005932c....js:2	1.7 kB	12.19 s
execute	200	fetch	vendors.005932c....js:2	35.0 kB	24.39 s
json	200	fetch	vendors.005932c....js:2	209 B	74 ms
json	200	fetch	vendors.005932c....js:2	209 B	76 ms
sortColumn	200	fetch	vendors.005932c....js:2	538 B	356 ms
json	200	fetch	vendors.005932c....js:2	209 B	55 ms

Name	Status	Type	Initiator	Size	Time
execute	200	fetch	vendors.005932c....js:2	4.6 kB	561 ms
execute	200	fetch	vendors.005932c....js:2	2.8 kB	555 ms
execute	200	fetch	vendors.005932c....js:2	7.0 kB	675 ms
execute	200	fetch	vendors.005932c....js:2	706 B	236 ms
execute	200	fetch	vendors.005932c....js:2	716 B	232 ms
execute	200	fetch	vendors.005932c....js:2	716 B	221 ms
execute	200	fetch	vendors.005932c....js:2	715 B	225 ms
execute	200	fetch	vendors.005932c....js:2	9.3 kB	292 ms
execute	200	fetch	vendors.005932c....js:2	1.7 kB	242 ms
execute	200	fetch	vendors.005932c....js:2	35.0 kB	850 ms
sortColumn	200	fetch	vendors.005932c....js:2	538 B	354 ms

调优思路



理解业务，选择需求

统计已经激活数字钱包用户数

- channel_id 代表渠道
- flag in (1, 257) 表示用户属于正常激活状态

```
SELECT COUNT(DISTINCT ba.uid) AS distinct_uid_count
FROM bank_account ba -- 银行账户表
LEFT JOIN user_register b ON ba.uid = b.uid -- 用户注册表
LEFT JOIN user_info a ON ba.uid = a.uid -- 用户个人信息表
WHERE ba.channel_id != 10004 -- 数字钱包服务
AND ba.flag IN (1, 257); -- 用户没有被封禁
```

无意义的关联

```
SELECT COUNT(DISTINCT ba.uid) AS distinct_uid_count
FROM bank_account_tab ba
WHERE ba.channel_id != 10004 -- 数字钱包服务
AND ba.flag IN (1, 257); -- 用户没有被封禁
```

理解业务，选择需求

统计非自营渠道的支付订单数

- 筛选字段存在JSON字符串中
- 枚举值type=4表示自营
- 枚举值type允许为空，也代表非自营

```
SELECT COUNT(1) AS cnt
FROM `order`
WHERE get_json_object(info, '$.type')!='4'
```

这样筛选对不对?

```
WHERE get_json_object(info, '$.type')!='4'
OR
get_json_object(info, '$.type') IS NULL
OR
get_json_object(info, '$.type’) = ''
```

利用日志

统计累计31天的留存

```
WITH new_activate_active_retention AS (
    SELECT t2.day AS dt,'new_activate_active_retention' AS retention_type,user_num, user_num_0d
    -- ...
    , user_num_31d
    FROM
    (
        SELECT
            t1.day
            ,COUNT(DISTINCT user_id) AS user_num
            ,COUNT(IF(INSTR(active_date_trace,day) > 0,1,NULL)) AS user_num_0d
,COUNT(IF(DATE_ADD(day,1)<='2023-09-01' AND INSTR(active_date_trace , DATE_ADD(day,0))+INSTR(active_date_trace , DATE_ADD(day,1))>0,1,NUL
-- 持续计算user_num_{n}d, 直到n=31
,COUNT(IF(DATE_ADD(day,31)<='2023-09-01' AND INSTR(active_date_trace , DATE_ADD(day,0))+INSTR(active_date_trace , DATE_ADD(day,1))+INSTR(
        FROM
        (
            SELECT
                user_id
                ,first_activate_date AS day
                ,active_date_trace
            FROM
            (
                SELECT new.uid AS user_id,new.first_activate_date,t.active_date_trace
                FROM
                (
                    SELECT uid, FROM_unixtime(first_activate_time, 'yyyy-MM-dd') AS first_activate_date
                    FROM user_info
                    WHERE FROM_UNIXTIME(first_activate_time, 'yyyy-MM-dd') >= '2022-10-01'
```

利用日志

统计累计31天的留存

```
# 放大executor内存
--conf spark.driver.memory=10g \
--conf spark.executor.instances=100 \
--conf spark.executor.memory=20g \
--conf spark.executor.cores=2 \
--conf spark.executor.memoryOverhead=8g
```

java.lang.OutOfMemoryError

```
# 关闭Spark AQE功能
--conf spark.sql.adaptive.enabled=false
```

274	pool-3-thread-2

```
scala.collection.immutable.StringLike.stripMargin$(StringLike.scala:185)
scala.collection.immutable.StringOps.stripMargin(StringOps.scala:33)
org.apache.spark.sql.catalyst.expressions.codegen.Block.toString(javaCode.scala:143)
org.apache.spark.sql.catalyst.expressions.codegen.Block.toString$(javaCode.scala:142)
org.apache.spark.sql.catalyst.expressions.codegen.EmptyBlock$.toString(javaCode.scala:306)
java.lang.String.valueOf(String.java:2994)
java.lang.StringBuilder.append(StringBuilder.java:131)
org.apache.spark.sql.catalyst.expressions.codegen.CodegenContext.$anonfun$evaluateSubExprEl
```

利用等价重写思想

统计电商收单笔数

```
SELECT COUNT(1)
FROM a t1
LEFT OUTER JOIN b t2
  ON t1.transaction_id = t2.transaction_id
WHERE t2.extinfo IS NULL;
```

先关联、再过滤

```
SELECT COUNT(1)
FROM a t1
LEFT OUTER JOIN (SELECT *
                 FROM b
                 WHERE extinfo IS NULL) t2
  ON t1.transaction_id = t2.transaction_id;
```

先过滤、再关联

熟知规则，利用规则

数据脱敏和字段拆解

- 敏感信息以PROTOBUF格式存储
- 先从PROTOBUF转换为JSON
- 根据需要将数据扁平处理

```
INSERT OVERWRITE TABLE result
-- 将extinfo转换为JSON字符串，取支付渠道id、银行卡id、卡指纹等
SELECT get_json_object(pb_to_json(get_json_object(`data`, ‘$.extinfo’), ‘class’),
‘$.info.channel_id‘)
            ,get_json_object(pb_to_json(get_json_object(`data`, ’$.extinfo’), ‘class’),
‘$.bank.id_no‘)
            ,get_json_object(pb_to_json(get_json_object(`data`, '$.extinfo'), 'class'),
'$.bank.fingerprint')
FROM order;
```

每行执行几次操作?

```
INSERT OVERWRITE TABLE result
SELECT get_json_object(info, ‘$.info.channel_id‘)
            ,get_json_object(info, ’$.bank.id_no‘)
            ,get_json_object(info, ’$.bank.fingerprint’)
-- extinfo转为JSON字符串的子查询
FROM (SELECT pb_to_json(get_json_object(`data`, ‘$.extinfo’), ‘class’) AS info
        FROM order) t;
```

转换为子查询有效果吗?

熟知规则，利用规则

数据脱敏和字段拆解

- 敏感信息以PROTOBUF格式存储
- 先从PROTOBUF转换为JSON
- 根据需要将数据扁平处理

```
INSERT OVERWRITE TABLE result
SELECT get_json_object(info, ‘$.info.channel_id‘)
            ,get_json_object(info, ’$.bank.id_no‘)
            ,get_json_object(info, ’$.bank.fingerprint’)
FROM (SELECT pb_to_json(get_json_object(`data`, ‘$.extinfo’), ‘class’) AS info -- 将extinfo转为JSON字符串
                        ,RAND() AS random_key -- 定义返回0-1之间随机数的字段
            FROM order) t
WHERE random_key < 2; -- 外层查询调用，且条件恒为TRUE
```

合并列优化规则

$$\Pi_{L1}(\Pi_{L2}(\ldots(\Pi_{Ln}(E))\ldots)) = \Pi_{L1}(E)$$

《大数据SQL优化 原理与实践》

开源社 kaiyuanshe

POWERDATA

您代表组织的LOGO
如有可在“设计-母版”放置

开源社 kaiyuanshe

POWERDATA

您代表组织的LOGO
如有可在“设计-母版”放置

PowerData社区公众号
扫码即可加入社区

开源社
kaiyuanshe

THANKS

Q&A

陈鹤

Shopee大数据专家