



第十一届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2020

架构革新 高效可控



北京国际会议中心 | 2020/12/21-12/23

大道相通，得鱼忘筌-从Oracle的SQL优化 到MySQL的SQL优化

巩飞，云和恩墨
2020/12/23



概述:

- 在国内，Oracle为王的时代，DBA们探索总结出很多SQL优化要点，现在MySQL成为新的王者，这些要点还有价值吗？我在这里将和大家探究Oracle到MySQL优化SQL的同与不同。比如，Oracle与MySQL的引擎模型、元数据模型、统计信息模型、执行计划模型、锁模型对比分析，提炼共通的SQL优化要点和原则，又考虑差异调整执行层面的细节。



巩飞 (Morinson)

云和恩墨应用架构产品部总经理，**SQM**产品经理

2002年工作至今，搞过开发、架构、运维等，如今专注产品，都是围绕着数据库这个领域。经历了两层架构时代关系型数据库技术的蓬勃发展、三层架构时代关系型数据库技术的砥砺前行、互联网+时代数据库技术面临的诸多挑战、一直到现在百家争鸣的数据库新时代。作为数据领域的老兵，很高兴能继续奋战在一线，和大家一起学习成长，乐在其中。

擅长场景化的**SQL**质控解决方案、**Oracle**、**MySQL**、内存数据库、**GoldenGate**，对**DB2**、**SQL Server**、**PostgreSQL**、**OceanBase**等数据库也有所了解。

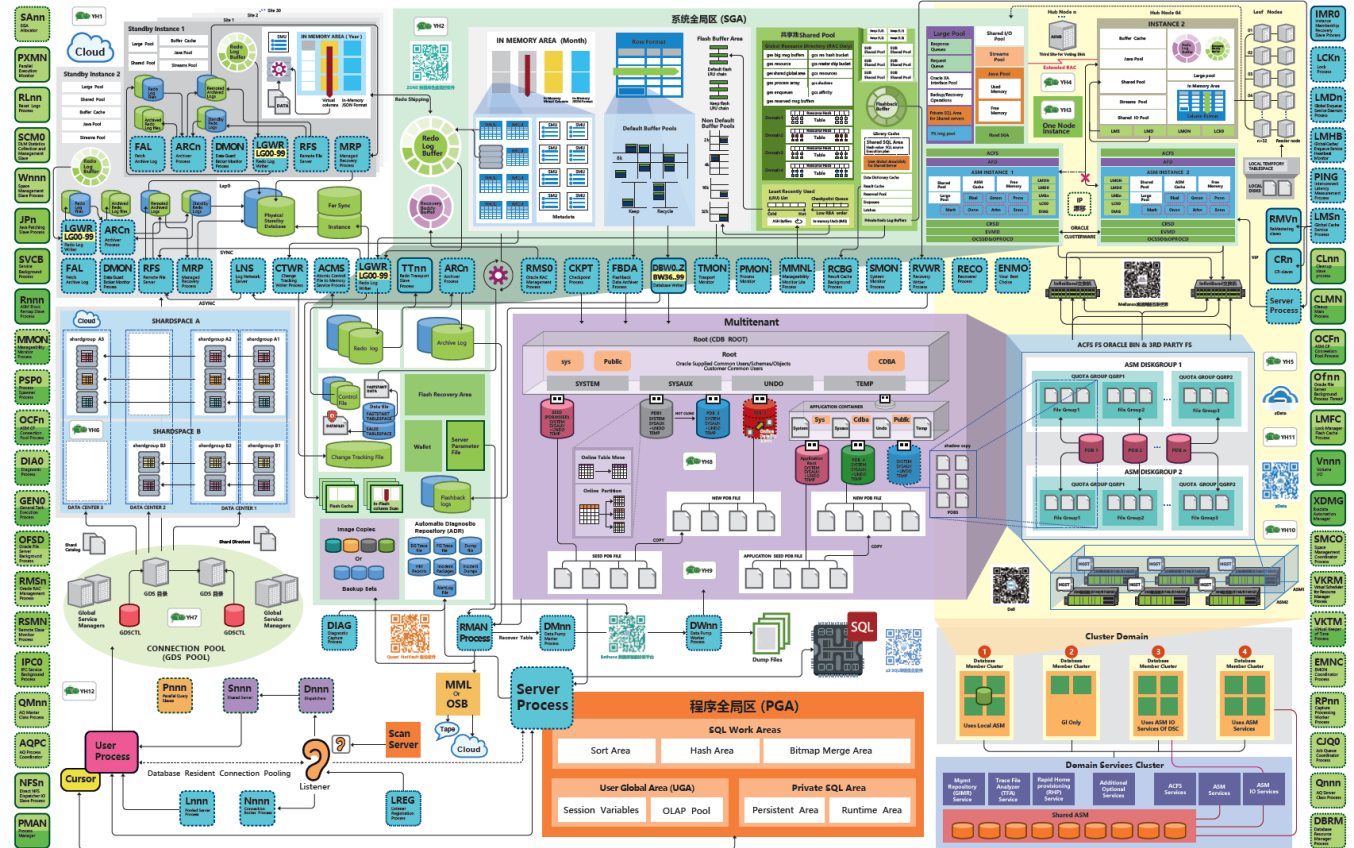


Oracle的体系结构



架构革新 高效可控
第十一届中国数据库技术大会
DATABASE TECHNOLOGY CONFERENCE CHINA 2020

Oracle Database 12c R2 Architecture Oracle数据库12c R2体系结构图



DTCC
2020

云和恩墨
YUNHE ENMO
云和恩墨(北京)信息技术有限公司
YUNHE ENMO (BEIJING) INFORMATION TECHNOLOGY CO., LTD.
4006608755 www.enmotech.com

北京分公司
上海办事处
广州分公司
成都办事处
南京分公司
贵阳办事处

服务



产品



代理 Oracle 数据库

代理 Oracle 中间件

代理 Oracle 12.2 的部署

代理 Oracle 12.2 的升级

代理 Oracle 12.2 的迁移

代理 Oracle 12.2 的备份

代理 Oracle 12.2 的恢复

代理 Oracle 12.2 的运维

代理 Oracle 12.2 的测试

代理 Oracle 12.2 的培训

代理 Oracle 12.2 的咨询

代理 Oracle 12.2 的集成

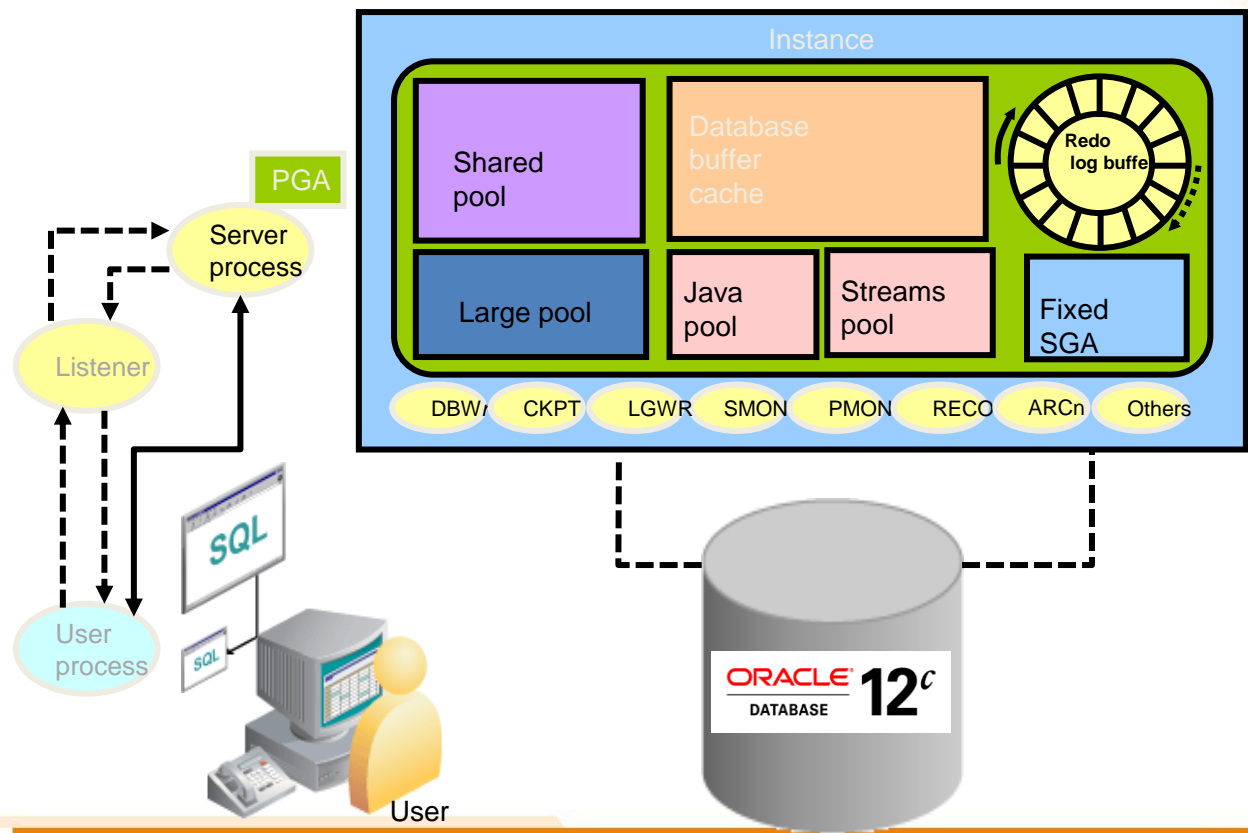
代理 Oracle 12.2 的定制

代理 Oracle 12.2 的优化

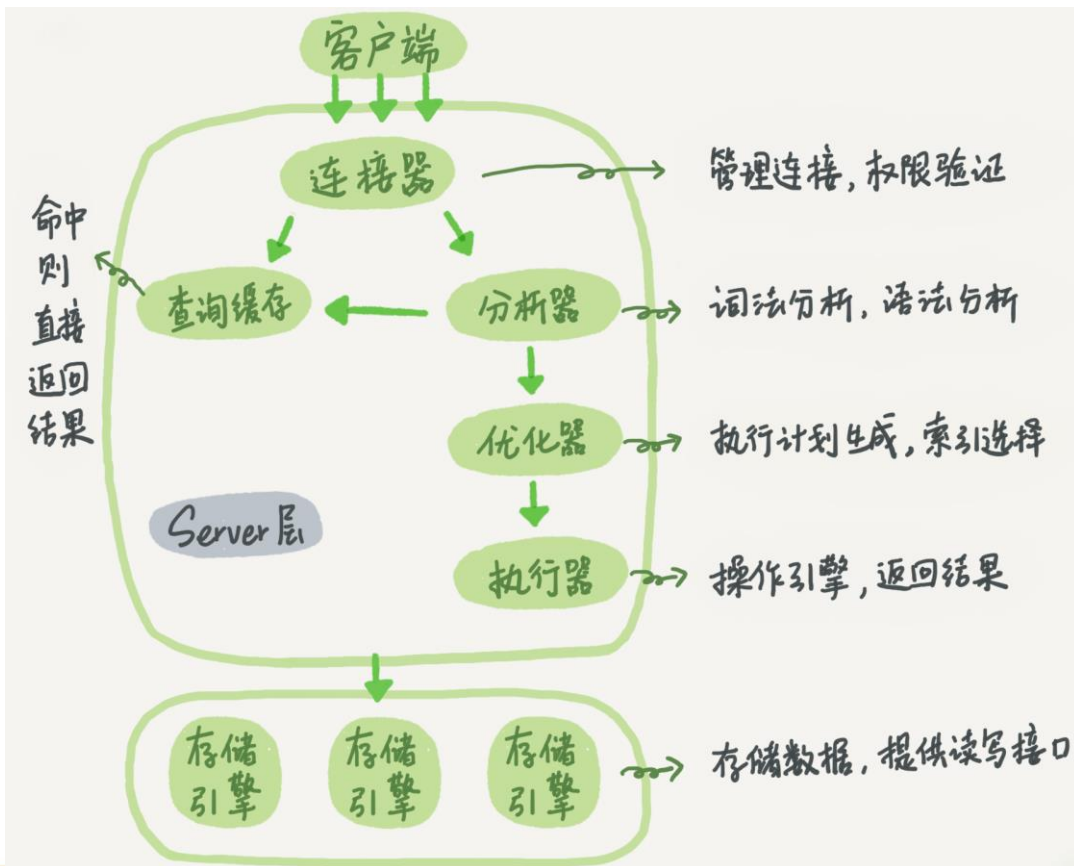
代理 Oracle 12.2 的迁移

chinaUnix
ITPUB

Oracle的体系结构-精简

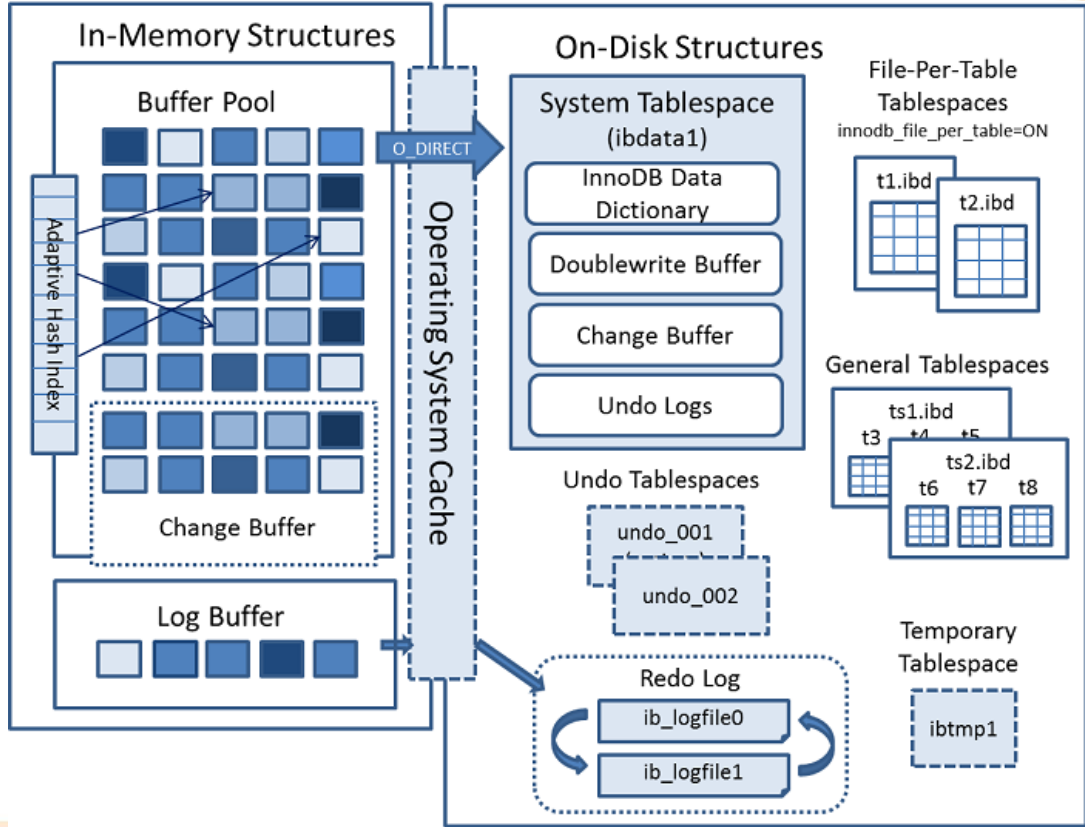


MySQL的体系结构-精简



注：此图引用自林晓斌老师的“MySQL实战45讲”

MySQL的体系结构-InnoDB架构

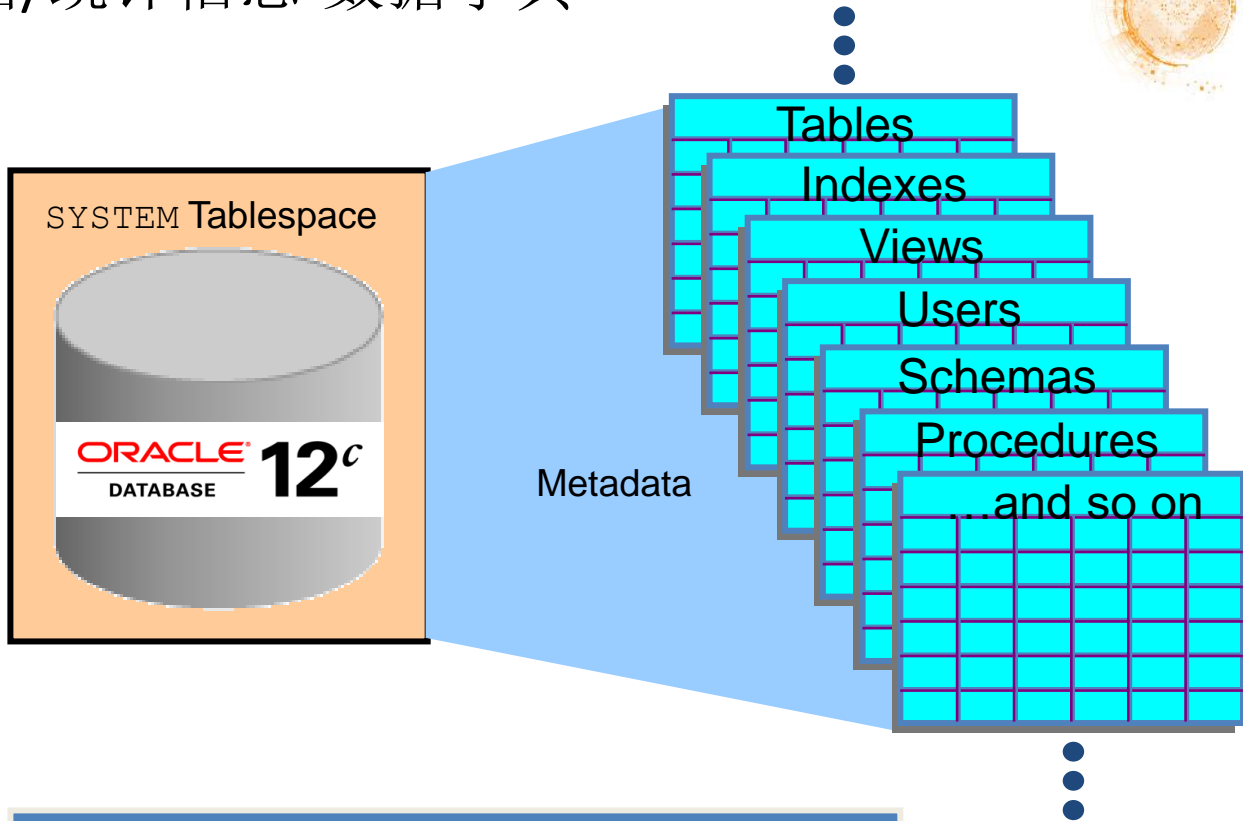


SQL的大处理环节类似，细节又很不同

MySQL的SQL解析器、优化器更轻量，更简单

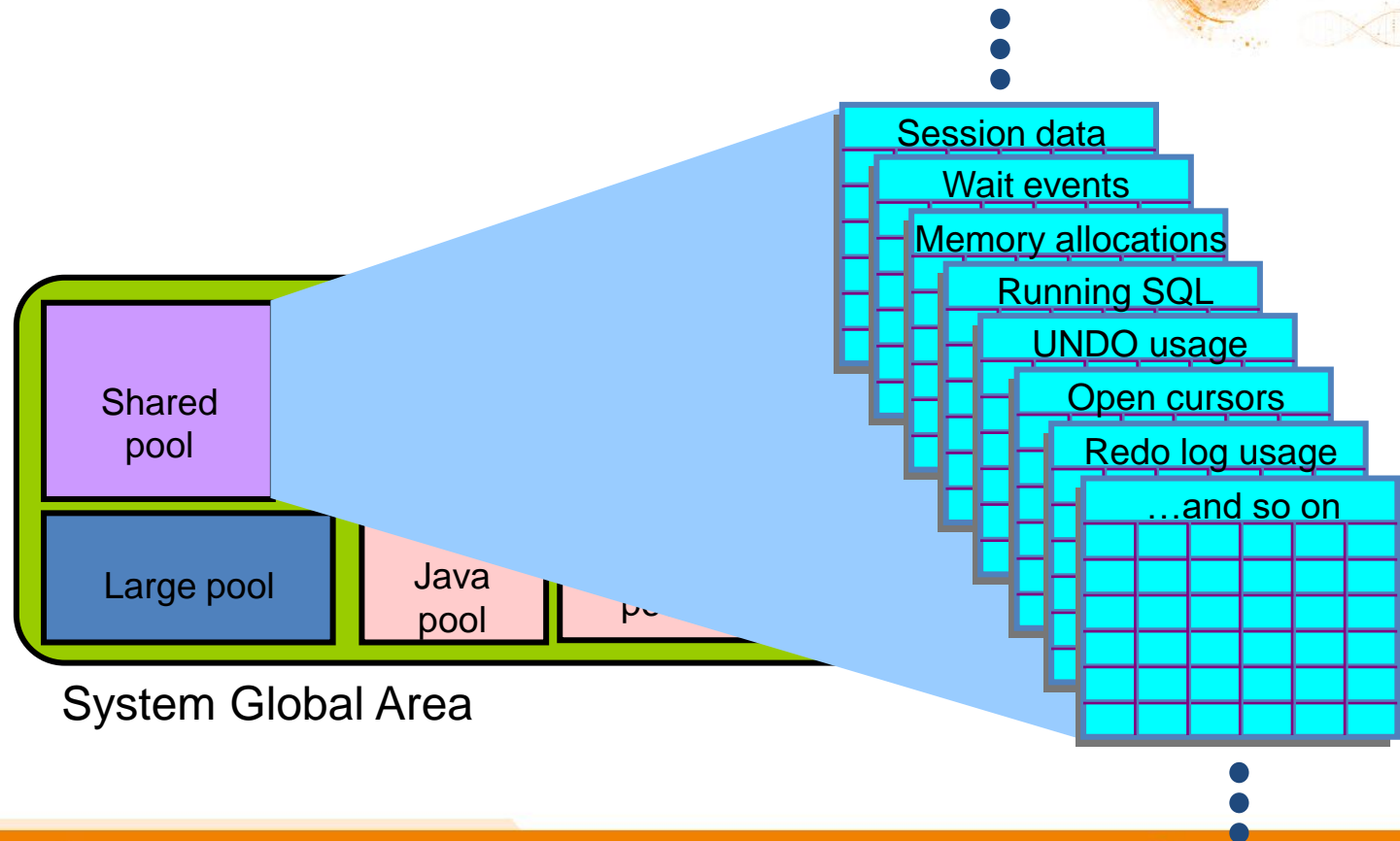
MySQL查询缓存的命中风险很大，特别是OLTP

Oracle元数据/统计信息-数据字典



```
SELECT * FROM dictionary;
```

Oracle元数据/统计信息-动态性能视图



MySQL元数据/统计信息-INFORMATION_SCHEMA

架构革新 © 高效可控
第十一届中国数据库技术大会
DATABASE TECHNOLOGY CONFERENCE CHINA 2020

```
root@database-one 11:56: [(none)]> use information_schema;  
Database changed  
root@database-one 11:56: [information_schema]> show tables;
```

```
+-----+  
| Tables_in_information_schema |  
+-----+  
| CHARACTER_SETS              |  
| COLLATIONS                  |  
| COLLATION_CHARACTER_SET_APPLICABILITY |  
| COLUMNS                    |  
| COLUMN_PRIVILEGES           |  
| ENGINES                     |  
| EVENTS                      |  
| FILES                       |  
| GLOBAL_STATUS               |  
| GLOBAL_VARIABLES           |  
| KEY_COLUMN_USAGE            |  
| OPTIMIZER_TRACE             |  
| PARAMETERS                  |  
| PARTITIONS                  |  
| PLUGINS                     |  
| PROCESSLIST                 |  
| PROFILING                   |  
| REFERENTIAL_CONSTRAINTS     |  
| ROUTINES                    |  
| SCHEMATA                    |  
| SCHEMA_PRIVILEGES           |  
| SESSION_STATUS              |  
| SESSION_VARIABLES           |  
| STATISTICS                  |  
| TABLES                     |  
| TABLESPACES                |  
| TABLE_CONSTRAINTS          |  
| TABLE_PRIVILEGES           |  
| TRIGGERS                    |  
| USER_PRIVILEGES             |  
| VIEWS                       |
```

```
TABLE_PRIVILEGES  
TRIGGERS  
USER_PRIVILEGES  
VIEWS  
INNODB_LOCKS  
INNODB_TRX  
INNODB_SYS_DATAFILES  
INNODB_FT_CONFIG  
INNODB_SYS_VIRTUAL  
INNODB_CMP  
INNODB_FT_BEING_DELETED  
INNODB_CMP_RESET  
INNODB_CMP_PER_INDEX  
INNODB_CMPMEM_RESET  
INNODB_FT_DELETED  
INNODB_BUFFER_PAGE_LRU  
INNODB_LOCK_WAITS  
INNODB_TEMP_TABLE_INFO  
INNODB_SYS_INDEXES  
INNODB_SYS_TABLES  
INNODB_SYS_FIELDS  
INNODB_CMP_PER_INDEX_RESET  
INNODB_BUFFER_PAGE  
INNODB_FT_DEFAULT_STOPWORD  
INNODB_FT_INDEX_TABLE  
INNODB_FT_INDEX_CACHE  
INNODB_SYS_TABLESPACES  
INNODB_METRICS  
INNODB_SYS_FOREIGN_COLS  
INNODB_CMPMEM  
INNODB_BUFFER_POOL_STATS  
INNODB_SYS_COLUMNS  
INNODB_SYS_FOREIGN  
INNODB_SYS_TABLESTATS
```

```
+-----+  
61 rows in set (0.00 sec)
```

```
root@database-one 11:56: [information_schema]>
```

元数据方面都比较全面，Oracle更丰富

基础的表、列、索引、统计信息都有，
满足常规SQL分析需要

Oracle提供了统一的元数据视图，MySQL
Server层的元数据统一，但存储插件相关的
由插件提供，比如InnoDB

Oracle执行计划

```
select * from dept,emp where emp.deptno=dept.deptno
```

```
Plan hash value: 844388907
```

Id	Operation	Name	E-Row
0	SELECT STATEMENT		
1	MERGE JOIN		
2	TABLE ACCESS BY INDEX ROWID	DEPT	
3	INDEX FULL SCAN	PK_DEPT	
* 4	SORT JOIN		
5	TABLE ACCESS FULL	EMP	

```
Query Block Name / Object Alias (identified by operation id):
```

```
1 - SEL$1
2 - SEL$1 / DEPT@SEL$1
3 - SEL$1 / DEPT@SEL$1
5 - SEL$1 / EMP@SEL$1
```

```
Outline Data
```

```
/*+
  BEGIN_OUTLINE_DATA
  IGNORE_OPTIM_EMBEDDED_HINTS
  OPTIMIZER_FEATURES_ENABLE('11.2.0.4')
  DB_VERSION('11.2.0.4')
  OPT_PARAM('_optimizer_extended_cursor_sharing'
  OPT_PARAM('_optimizer_extended_cursor_sharing_r
  OPT_PARAM('_optimizer_adaptive_cursor_sharing'
  OPT_PARAM('_optimizer_use_feedback' 'false')
  ALL_ROWS
  OUTLINE_LEAF(@"SEL$1")
  INDEX(@"SEL$1" "DEPT"@"SEL$1" ("DEPT"."DEPTNO"))
```

```
OUTLINE_LEAF(@"SEL$1")
INDEX(@"SEL$1" "DEPT"@"SEL$1" ("DEPT"."DEPTNO"))
FULL(@"SEL$1" "EMP"@"SEL$1")
LEADING(@"SEL$1" "DEPT"@"SEL$1" "EMP"@"SEL$1")
USE_MERGE(@"SEL$1" "EMP"@"SEL$1")
END_OUTLINE_DATA
```

```
*/
```

```
Predicate Information (identified by operation id):
```

```
4 - access("EMP"."DEPTNO"="DEPT"."DEPTNO")
   filter("EMP"."DEPTNO"="DEPT"."DEPTNO")
```

```
Column Projection Information (identified by operation id):
```

```
1 - "DEPT"."DEPTNO"[NUMBER,22], "EMP"."DEPTNO"[NUMBER,22], "DEPT"."LOC"[VARCHAR2,13],
   "DEPT"."DNAME"[VARCHAR2,14], "EMP"."EMPNO"[NUMBER,22], "EMP"."ENAME"[VARCHAR2,10], "EMP"."JOB"[VARCHAR2,9],
   "EMP"."MGR"[NUMBER,22], "EMP"."HIREDATE"[DATE,7], "EMP"."SAL"[NUMBER,22], "EMP"."COMM"[NUMBER,22]
2 - "DEPT"."DEPTNO"[NUMBER,22], "DEPT"."DNAME"[VARCHAR2,14], "DEPT"."LOC"[VARCHAR2,13]
3 - "DEPT".ROWID[ROWID,10], "DEPT"."DEPTNO"[NUMBER,22]
4 - (#keys=1) "EMP"."DEPTNO"[NUMBER,22], "EMP"."EMPNO"[NUMBER,22], "EMP"."ENAME"[VARCHAR2,10],
   "EMP"."JOB"[VARCHAR2,9], "EMP"."MGR"[NUMBER,22], "EMP"."HIREDATE"[DATE,7], "EMP"."SAL"[NUMBER,22],
   "EMP"."COMM"[NUMBER,22]
5 - "EMP"."EMPNO"[NUMBER,22], "EMP"."ENAME"[VARCHAR2,10], "EMP"."JOB"[VARCHAR2,9],
   "EMP"."MGR"[NUMBER,22], "EMP"."HIREDATE"[DATE,7], "EMP"."SAL"[NUMBER,22], "EMP"."COMM"[NUMBER,22],
   "EMP"."DEPTNO"[NUMBER,22]
```

```
Note
```

```
-----
- Warning: basic plan statistics not available. These are only collected when:
  * hint 'gather_plan_statistics' is used for the statement or
  * parameter 'statistics_level' is set to 'ALL', at session or system level
```

```
73 rows selected.
```


MySQL执行计划



```
root@database-one 13:06: [gfctest]> explain select * from dept,emp where emp.deptno=dept.deptno;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	dept	NULL	ALL	PRIMARY	NULL	NULL	NULL	3	100.00	NULL
1	SIMPLE	emp	NULL	ALL	NULL	NULL	NULL	NULL	6	16.67	Using where; Using join buffer (Block Nested Loop)

```
2 rows in set, 1 warning (0.00 sec)
```

```
root@database-one 13:06: [gfctest]> show warnings;
```

Level	Code	Message
Note	1003	/* select#1 */ select `gfctest`.`dept`.`deptno` AS `deptno`,`gfctest`.`dept`.`deptname` AS `deptname`,`gfctest`.`dept`.`adress` AS `adress`,`gfctest`.`emp`.`eno` AS `eno`,`gfctest`.`emp`.`ename` AS `ename`,`gfctest`.`emp`.`age` AS `age`,`gfctest`.`emp`.`sal` AS `sal`,`gfctest`.`emp`.`hiredate` AS `hiredate`,`gfctest`.`emp`.`deptno` AS `deptno` from `gfctest`.`dept` join `gfctest`.`emp` where (`gfctest`.`emp`.`deptno` = `gfctest`.`dept`.`deptno`)

1 row in set (0.00 sec)

- 访问方法
 - 是否以最好的方式访问数据？扫描？索引查找？
- 联接顺序
 - 是否以正确的顺序联接各表以便尽早尽多地消除数据？
- 联接类型
 - 是否使用了正确的联接类型？
- 分区修剪
 - 执行过分区修剪吗？是否消除了足够多的数据？
- 并行度
- 基数
 - 每个对象是否生成正确的行数？

执行计划-Oracle: 访问方法—取数据



访问方法	解释
全表扫描	读取表中所有行并过滤掉那些不符合 WHERE 子句谓词的行。用于索引、DOP 集等
按 ROWID 访问表	ROWID 指定含有所需行的数据文件和数据块以及该行在该块中的位置。当在索引或 WHERE 子句中提供 rowid 时使用
索引唯一扫描	将只返回一行。当语句中包含 UNIQUE 或 PRIMARY KEY 约束条件时使用，这些约束条件用于保证只访问一行
索引范围扫描	访问相邻索引项，可返回多个 ROWID 值。与等式一起用于非唯一索引，或与范围谓词一起用于唯一索引（<.>、between 等）
索引跳过扫描	如果前导列中只有很少的不同值，而非前导列中有许多不同的值，则跳过索引的前导部分，使用其余有用的部分
完整索引扫描	处理索引的所有叶块，但只有经过足够多的分支块才能找到第 1 个叶块。当所有需要的列都位于索引中且 order by 子句与索引结构匹配，或者排序合并联接已完成时，即可使用
快速完整索引扫描	扫描索引中的所有块，用来在所有需要的列都在索引中时代替 FTS。使用多块 IO，可以并行运行
索引联接	散列联接多个索引，这些索引一起包含有查询中引用的所有表列。不会消除排序操作
位图索引	使用键值位图和映射函数，映射函数可将每个比特的位置转换成一个 rowid。可以有效地合并对应于 WHERE 子句中的多个条件的索引

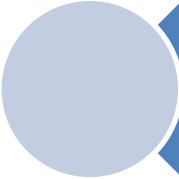


- 嵌套循环联接-Nested Loops Join
- 哈希联接- Hash Join
- 排序合并联接- Sort merge Joing
- 笛卡尔联接—MERGE JOIN CARTESIAN
-

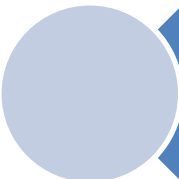
执行计划-MySQL: 访问方法与联接类型



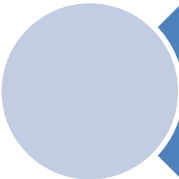
- 嵌套循环联接-
Nested Loops Join
- 全表扫描
- 索引访问



都有详细的执行计划步骤，这是
SQL性能分析的核心参考

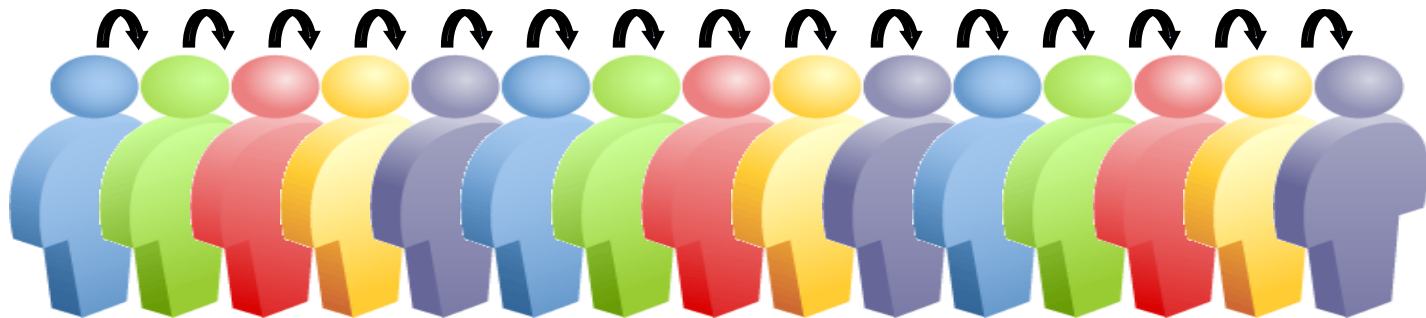


Oracle的执行计划中能提供更多访
问方法、联接类型



Oracle能提供更多的历史执行计划、
可进行执行计划之间的比对分析

- TM 锁(Table Lock)
- TX 锁 (Transaction Lock)
- HW 锁(High Watermark Lock)
- US 锁(Undo Segment Lock)
- TO 锁(Temporary Object Lock)
- CF 锁(Control File Lock)
- JO 锁(Job Queue Lock)
- SQ 锁(Sequence Cache)
-



Oracle锁-模式



模式	内部编号	模式值	描述(以DML为例来说明)
Null	KSQMNull	1	Null模式,不妨碍任何并发访问,主要用来作为Cache Invalidate的通知机制存在
SS	KSQMSS	2	SubShare模式,使用共享模式锁住一条记录
SX	KSQMSX	3	SubExclusive模式,使用独占模式锁住一条记录
S	KSQMS	4	共享模式
SSX	KSQMSSX	5	Share,SubExclusive,对表持有共享锁,对其中的记录持独占模式
X	KSQMX	6	Exclusive模式,对全表持独占模式

Oracle锁-模式兼容性



请求/占用	Null	SS	SX	S	SSX	X
Null	Yes	Yes	Yes	Yes	Yes	Yes
SS	Yes	Yes	Yes	Yes	Yes	No
SX	Yes	Yes	Yes	No	No	No
S	Yes	Yes	No	Yes	No	No
SSX	Yes	Yes	No	No	No	No
X	Yes	No	No	No	No	No

MySQL锁-按范围分类（InnoDB）



行锁

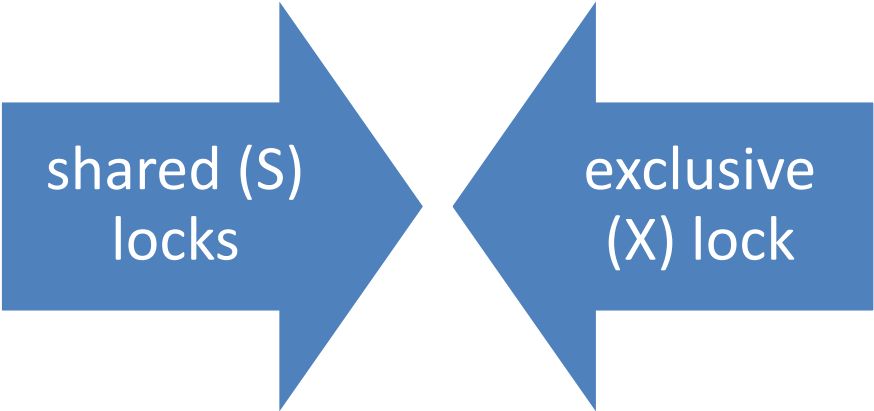
表锁

全局锁

MySQL锁-锁（ InnoDB ）

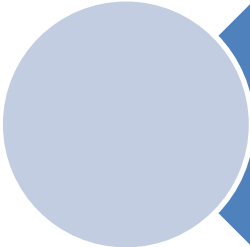


InnoDB行锁
分为两种

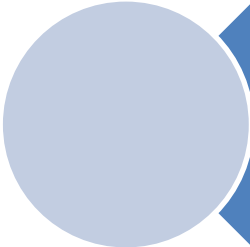


	X	IX	S	IS
X	Conflict	Conflict	Conflict	Conflict
IX	Conflict	Compatible	Conflict	Compatible
S	Conflict	Conflict	Compatible	Compatible
IS	Conflict	Compatible	Compatible	Compatible

注：表级锁兼容性

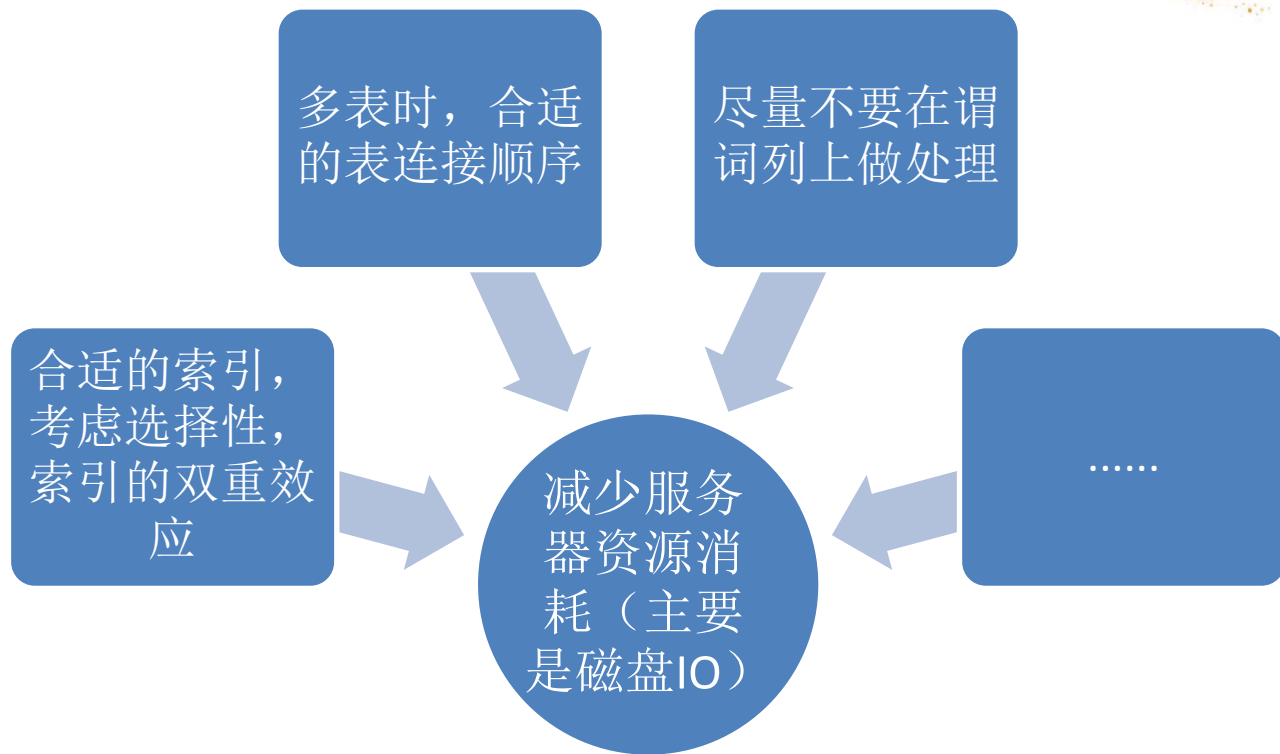


都有完善的锁机制，
能够支撑业务的并
发数据保护需求

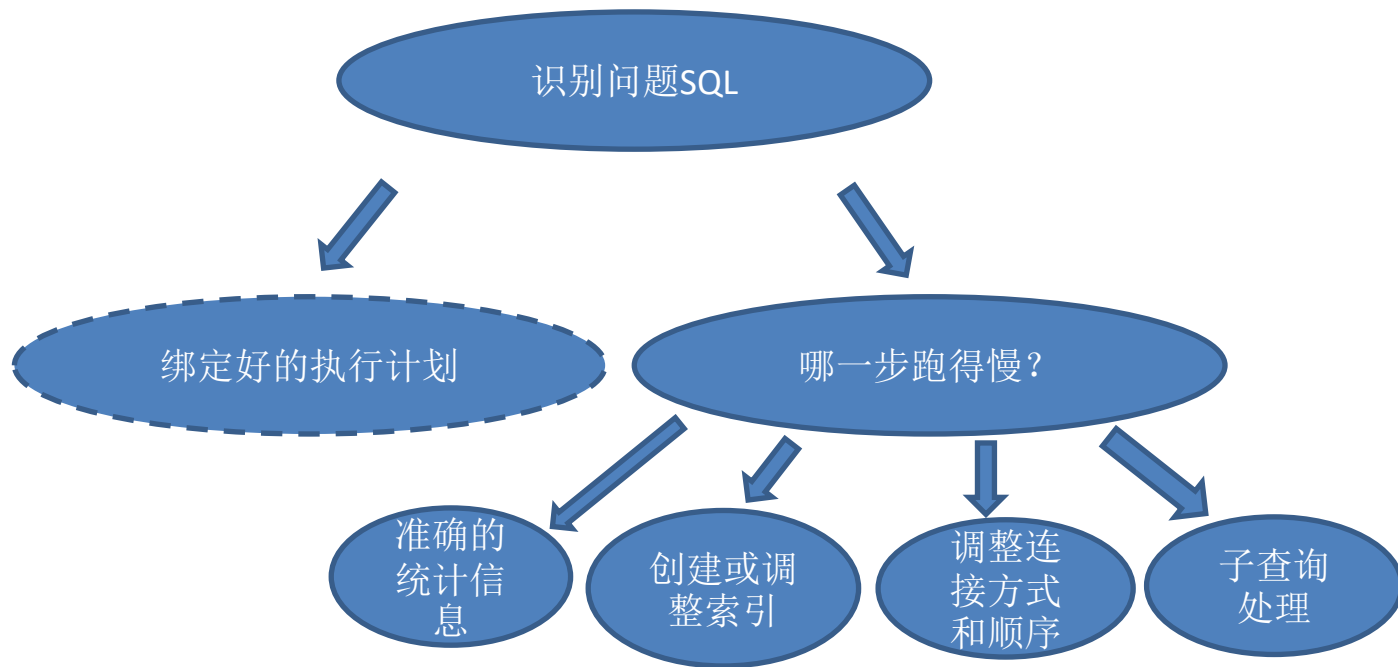


MySQL不同存储引
擎对锁的支持不同，
建议使用InnoDB

SQL优化的一般性原则



SQL优化的基本思路-Oracle与MySQL共通



虚框：Oracle有，MySQL无

如何高效的处理？方向展望



常见问题

- 缺乏适当索引列
- 条件上使用了运算符
- 条件上发生隐式转换
- 条件对应索引列不在复合索引第一位
- 条件对应索引列选择度不够高
- 统计信息不准确
-

专家优化
(手工)

自动识别
专家优化
(半自动)

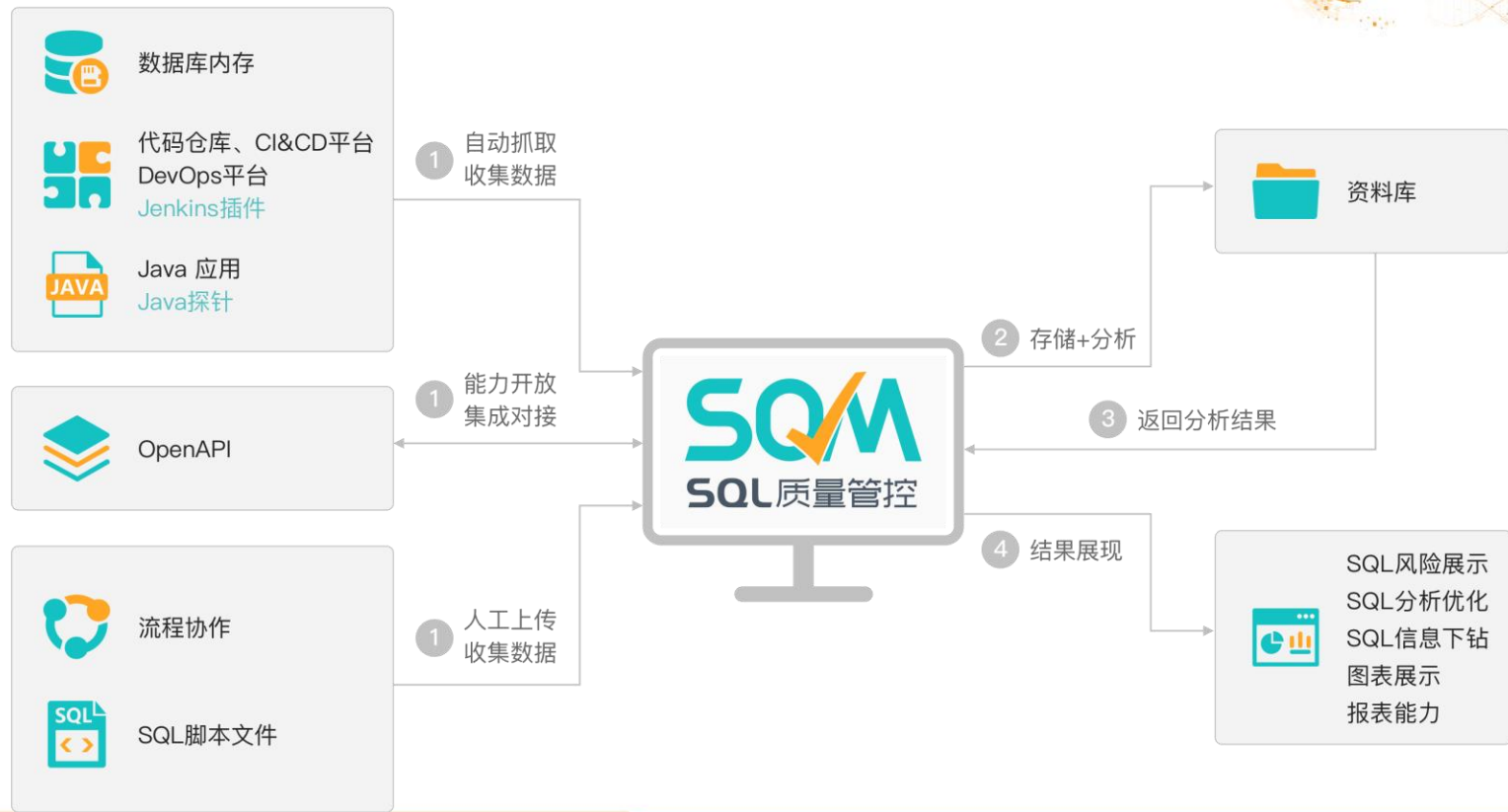
自动识别
自动处理
(全自动)

如何高效的处理？-知识库的建立和积累

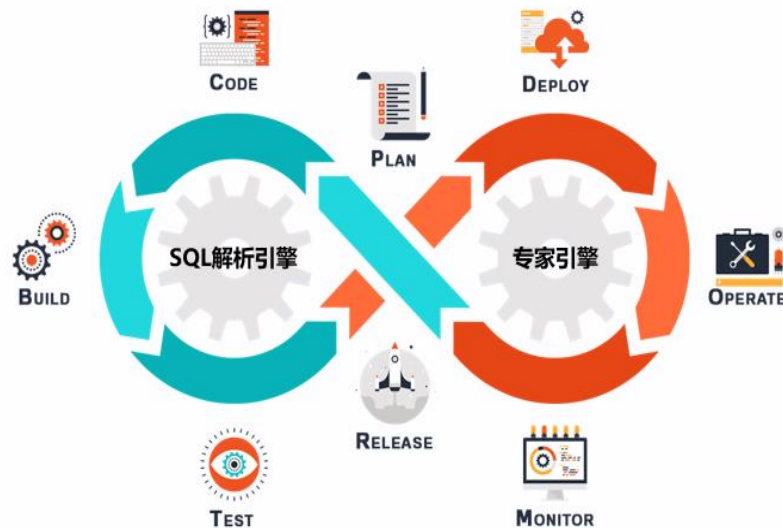
- 建立单位内部SQL相关的知识库
- 比如涵盖
 - SQL知识背景
 - SQL优化原则
 - SQL优化原理
 - SQL优化示例
 - SQL编写规范
- 随着业务发展不断积累更新



如何高效的处理？-产品化、工具化



如何高效的处理？-产品化、工具化



- ✓ 两大引擎，SQL解析引擎+专家引擎
- ✓ 覆盖应用全生命周期:开发、测试、上线发布、生产运行
- ✓ 内置的专家经验规则+语法树特征值、文本正则匹配的灵活自定义规则能力，全面覆盖用户SQL规范
- ✓ 支持各种开发语言构建的应用
- ✓ 已支持数据库Oracle、MySQL、DB2, GaussDB、Microsoft SQL Server 、OceanBase、PostgreSQL，各种国产数据库支持即将发布
- ✓ Jenkins插件、JVM插件，与DevOps集成，自动化SQL质控
- ✓ 强大的OpenAPI，方便用户快速将SQM能力与其它用户系统集成
- ✓ 支持不同规模、不同研发模式的十多种场景化SQL质量管控方案，适合的才是最好的
- ✓ 智能优化，能够对SQL进行全面分析后，给出精确的优化建议，比如创建索引等
- ✓ 支持X86架构、鲲鹏架构等

THANKS

