数据库设计开发规范

阿里云·数据库服务部

2010-07

反馈：aliyun-dba@list.alibaba-inc.com

目录

➢ Oracle

● 数据库整体设计规范（必读）

1．设计

1. 应用里面，多个数据库之间请不要通过 DBLINK 访问。
2. 请不要采用触发器。
3. 请不要使用视图和物化视图。
4. 请不要使用外键约束，如果数据存在外键关系，请在程序层面实现。
5. 请尽量不要使用 job，如果不得已必须使用，Job 的设计必须是可重复执行的。
6. 请尽量不要采用存储过程。
7. 应用必须具有自动重连的机制。但是又要避免每执行一条 SQL 语句就检查一下 DB 的可用性。

2．命名

a) 命名应使用富有意义的英文词汇，多个单词组成的，中间以下划线分割。
b) 命名只能使用英文字母，数字和下划线。
c) 命名避免使用 Oracle 保留字和系统关键字。
d) 命名长度以不超过 15 个字符为宜(避免超过 20)。
e) 命名全部采用小写，并且名称前后不能加引号。

● 数据库对象设计规范

1. 表

设计

a) 在设计时尽量包含两个日期字段:gmt_created(创建日期),gmt_modified(修改日期)且非空, 对表的记录进行更新的时候，必须包含对 gmt_modified 字段的更新。
b) 尽可能使用简单数据类型，不要使用类似数组或者嵌套表这种复杂类型。
c) 必须要有主键，且尽量不要使用有实际意义的字段做主键。
d) 需要 join 的字段，数据类型保持绝对一致。
e) 当表的字段数非常多时，可以将表分成两张表，一张作为条件查询表，一张作为详细内容表（主要是为了性能考虑）。
f) 当字段的类型为枚举型或布尔型时，建议使用 char(1)类型。

命名

a) 同一个模块的表尽可能使用相同的前缀，表名尽可能表达含义，例如：CRM_SAL_FUND_ITEM。
b) 字段命名应尽可能使用表达实际含义的英文单词或缩写，不要使用类似“VALUE1”这种无意义的字段名。
c) 布尔值类型的字段命名为 is+描述。如 member 表上表示是否为 enabled 的会员的字段命名为 IsEnabled。

字段类型

类型	规范
NUMBER(p,s)	固定精度数字类型
NUMBER	不固定精度数字类型，当不确定数字的精度时使用，PK 通常使用此类型
DATE	当仅需精确到秒时，选择 DATE 而不是 TIMESTAMP 类型
TIMESTAMP	扩展日期类型，不建议使用
VARCHAR2	变长字符串，最长 4000 个字节
CHAR	定长字符串，除非是 CHAR(1),否则不要使用
CLOB	当超过 4000 字节时使用，但是要求这个字段必须单独创建到一张表中，然后有 PK 与主表关联。此类型应该尽量控制使用

字段注释

a) 标准字段注释由一组"@"开头的标签+空格+文本组成。

以 MD_USER 表的部分字段为例：

Name	Type	Comments
PARTY_ID	VARCHAR2(20)	@desc 主键 ID
CORPORATION_ID	VARCHAR2(20)	@desc 用户所在公司 ID @fk md_corporation.party_id
STATUS	VARCHAR2(20)	@desc 状态 @values disable|enable: 未激活状态|激活状态
IS_PRI_ACCOUNT	CHAR(1)	@desc 是否为主账号。后台生成 UK 时使用 @values y|n:是帐号，非主帐号 @logic 一个公司内部，有且仅有一个主账号存在

b) 注释标签说明

标签名	中文含义	必填	备注
@desc	字段中文描述	Yes	
@fk	字段对应的外键字段		
@values	取值范围说明。多个值以"|"分隔		如此字段的值由系统自动生成，可忽略不书写。
@sample	数据范本		对于复杂数据格式，最好给一个数据范本。
@formula	计算公式		写明该字段由哪些字段以何种公式计算得到。
@logic	数据逻辑		简要写明该字段的数据是在何种业务规则下，如何变化的。
@redu	标识此字段冗余		
@depr	标识此字段已废弃		简要写明：废弃人 废弃日期 废弃原因

2. 索引

设计

a) Bitmap 索引通常不适合我们的环境。
b) 索引根据实际 SQL，由 DBA 创建。
c) 不要创建带约束的索引，所有的约束效果都通过显示创建约束然后再 using index 一个已经创建好的普通索引来实现。

命名

a) <table_name>_<column_name>_ind,各部分以下划线（_）分割。
b) 多单词组成的 column name，取前几个单词首字母，加末单词组成 column_name。如 sample 表 member_id 上的索引：sample_mid_ind。

3. 约束

设计

a) 主键最好是无意义的，由 Sequence 产生的 ID 字段，类型为 number，不建议使用组合主键。
b) 若要达到唯一性限制的效果，不要创建 unique index，必须显式创建普通索引和约束（pk 或 uk），即先创建一个以约束名命名的普通索引，然后创建一个约束，用 using index ...指定索引。
c) 当删除约束的时候，为了确保不影响到 index，最好加上 keep index 参数。
d) 主键的内容不能被修改。
e) 外键约束一般不在数据库上创建，只表达一个逻辑的概念，由程序控制。
f) 当万不得已必须使用外健的话，必须在外健列创建 INDEX。

命名

a) 主键约束： _pk 结尾，<table_name>_pk；
b) unique 约束：_uk 结尾，<table_name>_<column_name>_uk；
c) check 约束： _ck 结尾，<table_name>_<column_name>_ck；
d) 外键约束： _fk 结尾，以 pri 连接本表与主表，<table_name>_pri_<table_name>_fk；

4. SEQUENCE

命名

a) seq_<table_name>

5. 触发器

命名

a) <table_name>_A(After)B(Before)I(Insert)U(Update)D(Delete)_trg。
b) 若是用于同步的触发器以 sync 作为前缀：sync_<table_name>_trg。

6. 过程、函数、包

命名

a) 过程以 proc_开头，函数以 func_开头，包以 pkg_开头。
b) 变量命名约定：本地变量以 v_为前缀，参数以 p_为前缀，可以带_I(输入)，_O(输出)、_IO(输入输出）表示参数的输入输出类型。

● SQL 开发规范

一. 编码规范

1. 在代码中不允许出现任何 DDL 语句

a) DDL 语句一律由 DBA 编写并统一执行

2. 写 SQL 的时侯一定要使用绑定变量

a) 对于极少数情况下不使用绑定变量提高性能，使用之前一定要和 DBA 沟通

3. 写 SQL 的时候一定要给每个字段指定表名做前缀

a) 比如 select a.id,a.name from test a; 好处是一来带来性能的提升，二来可以避免一些错误的发生。

4. 在 sqlmap 中的变量，要用#号，而不要用$符号

a) 如#appid#。因为$name$ 是字面意义的替换，这种形式会有 SQL 注入的漏洞，而 #name# 是带类型的替换，不存在 SQL 注入的风险。

5. 请不要写 select * 这样的代码，指定需要的字段名

6. 避免在 where 子句中对字段施加函数

a) 通常，不允许在字段上添加函数或者表达式，这样将导致索引失效，如：

```
错误的写法：
select * from iw_account_log where to_char ( trans_dt, 'yyyy-mm-dd') = '2007-04-04';
select qty from product where p_id + 12 = 168;

正确的写法：
select * from iw_account_log
where trans_dt >= to_date ( '2007-04-04', 'yyyy-mm-dd') and trans_dt < to_date
( '2007-04-05', 'yyyy-mm-dd');
select qty from product where p_id = 168 - 12;
```

b) 如果是业务要求的除外，但需要在编写时咨询 DBA

c) 特别注意，当表连接时，用于连接的两个表的字段如果数据类型不一致，则必须在一边加上类型转换的函数，如

```
错误的写法（a.id 是 number 类型，而 b.operator_number 是 char 类型）：
select count  from adm_user a, adm_action_log b where a.id = b.operator_number
and a.username = '小钗';
```

正确的写法：

select count from adm_user a, adm_action_log b where to_char(a.id) = b.operator_number and a.username = '小钗';

select count from adm_user a, adm_action_log b where a.id = to_number(b.operator_number) and a.username = '小钗';

上面两种写法哪个正确？遇到这种情况时必须咨询 DBA！

7. 严格要求使用正确类型的变量，杜绝 oracle 做隐式类型转换的情况

a) 推荐在 sqlmap 的变量中指定变量的数据类型，如：select * from iw_user where iw_user_id = #userid:VARCHAR#

b) 其中，对于时间类型的字段，必须使用 TO_DATE 进行赋值（当前时间可直接用 sysdate 表示），不允许下列这些错误用法：

错误的写法（使用 date 类型的变量）：

```
select *
from iw_account_log
where trans_account = #transaccount:varchar#
   and trans_dt >= #dateBegin:date#
   and trans_dt < #dateEnd:date#
```

错误的写法（将 to_date 函数和数字进行算术运算）：

```
select *
from iw_account_log
where trans_account = #transaccount:varchar#
   and trans_dt >= to_date(#dateBegin:varchar#, 'yyyy-mm-dd hh24:mi:ss')
   and trans_dt < to_date(#dateBegin:varchar#, 'yyyy-mm-dd hh24:mi:ss') + 1
```

正确的写法：

```
select *
from iw_account_log
where trans_account = #transaccount:varchar#
  and trans_dt >= to_date(#dateBegin:varchar#, 'yyyy-mm-dd hh24:mi:ss')
  and trans_dt < to_date(#dateEnd:varchar#, 'yyyy-mm-dd hh24:mi:ss') /*或  trans_dt < sysdate */
```

c) 3、对于变量数据类型错误导致 SQL 严重性能问题的，按严重的编码错误 Bug 处理！

8. 全模糊查询无法使用 INDEX，应当尽可能避免

a) 比如：select * from table where name like '%jacky%';

9. 外连接的写法

a) 不推荐使用 ANSI 连接，如 inner join、left join、right join、full outer join，而推荐使用（+）来表示外连接

不推荐的写法：

```
select a.*, b.goods_title from iw_account_log a left join beyond_trade_base b on
a.TRANS_OUT_ORDER_NO = b.trade_no
where a.trans_code = '6003' and a.trans_account = #transacnt:varchar# and
a.trans_dt > to_date(...)
```

推荐的写法：

```
select a.*, b.goods_title from iw_account_log a, beyond_trade_base b
where a.TRANS_OUT_ORDER_NO = b.trade_no(+)  and a.trans_code = '6003'
 and a.trans_account = #transacnt:varchar# and a.trans_dt > to_date(...)
```

10. 表连接分页查询的使用

a) 包含排序逻辑的分页查询写法，必须是三层 select 嵌套：

错误的写法：

```
SELECT t1.*
 FROM (SELECT t.*, ROWNUM rnum
       FROM beyond_trade_base t
       WHERE seller_account = :1
        AND gmt_create >= TO_DATE (:2, 'yyyy-mm-dd')
        AND gmt_create < TO_DATE (:3, 'yyyy-mm-dd')
       ORDER BY gmt_create DESC) t1
 WHERE rnum >= :4 AND rnum < :5
```

正确的写法：

```
SELECT t2.*
  FROM (SELECT t1.*, ROWNUM rnum
        FROM (SELECT t.*
              FROM beyond_trade_base t
              WHERE seller_account = :1
               AND gmt_create >= TO_DATE (:2, 'yyyy-mm-dd')
               AND gmt_create < TO_DATE (:3, 'yyyy-mm-dd')
              ORDER BY gmt_create DESC) t1
              WHERE ROWNUM <= :4) t2
  WHERE rnum >= :5
```

b) 不包含排序逻辑的分页查询写法，则是两层 select 嵌套，但对 rownum 的范围指定仍然必须在不同的查询层次指定：

错误的写法：

```
SELECT t1.*
 FROM (SELECT t.*, ROWNUM rnum
       FROM beyond_trade_base t
       WHERE seller_account = :1
        AND gmt_create >= TO_DATE (:2, 'yyyy-mm-dd')
        AND gmt_create < TO_DATE (:3, 'yyyy-mm-dd')) t1
 WHERE rnum >= :4 AND rnum <= :5
```

正确的写法：

```
SELECT t1.*
 FROM (SELECT t.*, ROWNUM rnum
       FROM beyond_trade_base t
       WHERE seller_account = :1
        AND gmt_create >= TO_DATE (:2, 'yyyy-mm-dd')
        AND gmt_create < TO_DATE (:3, 'yyyy-mm-dd')
        AND ROWNUM <= :4) t1
 WHERE rnum >= :5
```

c) 注意下面两种写法的逻辑含义是不同的：

按创建时间排序（倒序），然后再取前 10 条：

```
SELECT t2.*
 FROM (SELECT t1.*, ROWNUM rnum
       FROM (SELECT t.*
             FROM sell_offer t
             WHERE owner_member_id = :1
               AND gmt_create >= TO_DATE (:2, 'yyyy-mm-dd')
               AND gmt_create < TO_DATE (:3, 'yyyy-mm-dd')
             ORDER BY gmt_create DESC) t1
       WHERE ROWNUM <= 10) t2
 WHERE rnum >= 1
```

随机取 10 条，然后在这 10 条中按照交易创建时间排序（倒序）：

```
SELECT t1.*
FROM (SELECT t.*, ROWNUM rnum
FROM beyond_trade_base t
WHERE seller_account = :1
AND gmt_create >= TO_DATE (:2, 'yyyy-mm-dd')
AND gmt_create < TO_DATE (:3, 'yyyy-mm-dd')
AND ROWNUM <= 10
```

```
ORDER BY gmt_create DESC) t1
WHERE rnum >= 1
```

d) 先连接后分页与先分页后连接

性能较差：

```
SELECT t2.*
FROM (SELECT t1.*, ROWNUM rnum
FROM (SELECT a.*, b.receive_fee
FROM beyond_trade_base a, beyond_trade_process b
WHERE a.trade_no = b.trade_no
AND a.seller_account = :1
AND a.gmt_create >= TO_DATE (:2, 'yyyy-mm-dd')
AND a.gmt_create < TO_DATE (:3, 'yyyy-mm-dd')
ORDER BY a.gmt_create DESC) t1
WHERE ROWNUM <= :4) t2
WHERE rnum >= :5
```

性能较好：

```
SELECT /*+ ordered use_nl(a,b) */
a.*, b.receive_fee
FROM (SELECT t2.*
FROM (SELECT t1.*, ROWNUM rnum
FROM (SELECT t.*
FROM beyond_trade_base t
WHERE seller_account = :1
AND gmt_create >= TO_DATE (:2, 'yyyy-mm-dd')
AND gmt_create < TO_DATE (:3, 'yyyy-mm-dd')
ORDER BY gmt_create DESC) t1
WHERE ROWNUM <= :4) t2
WHERE rnum >= :5) a,
beyond_trade_process b
WHERE a.trade_no = b.trade_no
```

后面这种写法的适用情况：

1)where 子句中的查询条件都是针对 beyond_trade_base 表的（否则得到的结果将不相同）

2）关联 beyond_trade_process 表时，用的是该表的主键或者唯一键字段（否则将改变结果集的条数）

11. Hint 的使用

a) sql 中的/*+ ordered use_nl(member offer)*/是 hint，用来确定 SQL 的执行计划，请在 DBA 确认后使用。

12. ″<>″、″!=″、″not in″、″exsits″和″not exists″的使用规范

a) 原则上一般禁止使用"<>"、"!="和"not in"，而应该转换成相应的"="和"in"查询条件

错误的写法：

```
select a.id,a.subject,a.create_type
from product
where status <> 'new'
  and owner_member_id = :1
```

正确的写法：

```
select a.id,a.subject,a.create_type
from product
where status in ('auditing','modified','service-delete','tbd','user-delete','wait-for-audit')
  and owner_member_id = :1
```

错误的写法：

```
select a.id,a.subject,a.create_type
from product
where create_type not in ('new_order','vip_add')
  and owner_member_id = :1
```

正确的写法：

```
select a.id,a.subject,a.create_type
from product
where create_type = 'cust_add'
  and owner_member_id = :1
```

b) 原则上不允许使用"exsits"和"not exists"查询，应转换成相应的"等连接"和外连接来查询

错误的写法：

```
select a.id
from company a
  where not exsits (select 1
                    from av_info_new b
                    where a.id = b.company_id
)
```

正确的写法：

```
select a.id
from company a,av_info_draft b
```

```
where a.id = b.company_id
  and b.company_id is null
```

错误的写法：

```
select count
from company a
where exsits (select 1
              from av_info_new b
              where a.id = b.company_id
)
```

正确的写法：

```
select count
from company a,av_info_draft b
where a.id = b.company_id
```

注：在通过等连接替换 exsits 的时候有一点需要注意，只有在一对一的时候两者才能较容易替换，如果是一对多的关系，直接替换后两者的结果会出现不一致情况。因为 exsits 是实现是否存在，他不 care 存在一条还是多条，而等连接时返回所关联上的所有数据。

c) 如有特殊需要无法完成相应的转换，必须在 DBA 允许的情况下使用"<>"、"!="、"not in"、"exsits"和"not exists"

13. 其它编写规范

a) 对表的记录进行更新的时候，必须包含对 gmt_modified 字段的更新，并且不要使用 dynamic 标记，如：

错误的写法：

```
update BD_CONTACTINFO
<dynamic prepend="set">
  .....
  <isNotNull prepend="," property="gmtModified">
    GMT_MODIFIED = #gmtModified:TIMESTAMP#
  </isNotNull>
</dynamic>
where ID = #id#
```

正确的写法（当然，这里更推荐直接更新为 sysdate）：

```
update BD_CONTACTINFO
set GMT_MODIFIED = #gmtModified:TIMESTAMP#
<dynamic>
```

```
 ......
</dynamic>
where ID = #id#
```

b) 不允许在 where 后添加 1=1 这样的无用条件，where 可以写在 prepend 属性里，如：

错误的写法：

```
select count      from BD_CONTRACT t where 1=1
<dynamic>
 ......
</dynamic>
```

正确的写法：

```
select count      from BD_CONTRACT t
<dynamic prepend="where">
 ......
</dynamic>
```

c) 对大表进行查询时，在 SQLMAP 中需要加上对空条件的判断语句，具体可在遇到时咨询 DBA，如：

性能上不保险的写法：

```
select count      from iw_user usr
<dynamic prepend="where">
 <isNotEmpty prepend="AND" property="userId">
  usr.iw_user_id = #userId:varchar#
 </isNotEmpty>
 <isNotEmpty prepend="AND" property="email">
  usr.email = #email:varchar#
 </isNotEmpty>
 <isNotEmpty prepend="AND" property="certType">
  usr.cert_type = #certType:varchar#
 </isNotEmpty>
 <isNotEmpty prepend="AND" property="certNo">
  usr.cert_no = #certNo:varchar#
 </isNotEmpty>
</dynamic>
```

性能上较保险的写法（防止那些能保证查询性能的关键条件都为空）：

```
select count      from iw_user usr
<dynamic prepend="where">
```

```
<isNotEmpty prepend="AND" property="userId">
 usr.iw_user_id = #userId:varchar#
</isNotEmpty>
<isNotEmpty prepend="AND" property="email">
 usr.email = #email:varchar#
</isNotEmpty>
<isNotEmpty prepend="AND" property="certType">
 usr.cert_type = #certType:varchar#
</isNotEmpty>
<isNotEmpty prepend="AND" property="certNo">
 usr.cert_no = #certNo:varchar#
</isNotEmpty>
<isEmpty property="userId">
 <isEmpty property="email">
  <isEmpty property="certNo">
   query not allowed
  </isEmpty>
 </isEmpty>
</isEmpty>
</dynamic>
```

另外，对查询表单的查询控制建议使用 web 层进行控制而不是客户端脚本（JAVASCRIPT/VBSCRIPT）

d) 聚合函数常见问题
   1) 不要使用 count(1)代替 count（*）
   2) count(column_name)计算该列不为 NULL 的记录条数
   3) count(distinct column_name)计算该列不为 NULL 的不重复值数量
   4) count()函数不会返回 NULL，但 sum()函数可能返回 NULL，可以使用 nvl(sum(qty),0)来避免返回 NULL

e) NULL 的使用
   1) 理解 NULL 的含义，是"不确定"，而不是"空"
   2) 查询时，使用 is null 或者 is not null
   3) 更新时，使用等于号，如：update tablename set column_name = null

f) STR2NUMLIST、STR2VARLIST 函数的使用：
   1) 适用情况：使用唯一值（或者接近唯一值）批量取数据时
   2) 编写规范：a 表必须放在 from list 的第一位，并且必须在 select 后加上下面的 hint

正确的写法：

```
select /+ ordered use_nl(a,b) */ b.
from TABLE(CAST(str2varlist(:1) as vartabletype)) a, beyond_trade_base b
where a.column_value = b.trade_no;
```

二. 格式规范

1．注释说明

a) 本注释说明主要用于 PL/SQL 程序及其它 SQL 文件，其它可作参考；
b) SQLPLUS 接受的注释有三种：

```
－－  这儿是注释
/ *  这儿是注释  * /
REM  这儿是注释
```

c) 开始注释，类似 JAVAK 中的开始注释，主要列出文件名，编写日期，版权说明，程序功能以及修改记录：

```
REM
REM  $Header: filename, version, created date,auther
REM
REM Copyright
REM
REM  FUNCTION
REM  function explanation
REM
REM  NOTES
REM
REM  MODIFIED  （yy/mm/dd）
REM    who   when   - for what, recently goes first
```

d) 块注释，如表注释，PROCEDURE 注释等，同 JAVA:

```
/*
 * This table is for TrustPass
 * mainly store the information
 * of TrustPass members
 */
```

注意: 在“/*”后应当另起一行，或与其后描述有间隔，否则在 SQLPLUS 中会有问题。

e) 单行注释，如列注释：

```
login_id      VARCHAR2(32) NOT NULL,  -- 会员标识
```

2．缩进

低级别语句在高级别语句后的，一般缩进 4 个空格：

```
DECLARE
    v_MemberId      VARCHAR2(32),
BEGIN
    SELECT admin_member_id INTO v_MemberId
      FROM company
      WHERE id = 10;
    DBMS_OUTPUT.PUT_LINE(v_MemberId);
END;
```

同一语句不同部分的缩进，如果为 sub statement，则通常为 2 个空格，如果与上一句某部分有密切联系的，则缩至与其对齐：

```
BEGIN
FOR v_TmpRec IN
(SELECT login_id,
                gmt_created, -- here indented as column above
                  satus
              FROM  member          -- sub statement
              WHERE site = 'china'
                AND country='cn' )
  LOOP
      NULL;
  END  LOOP;
END;
```

3．断行

a) 一行最长不能超过 80 字符
b) 同一语句不同字句之间
c) 逗号以后空格
d) 其他分割符前空格

```
SELECT  offer_name
     ||','
     ||offer_count as offer_category,
       id
  FROM category
  WHERE super_category_id_1 = 0;
```

● 附录：Oracle 关键字

ACCESS	DECIMAL	INITIAL	ON	START
ADD	NOT	INSERT	ONLINE	SUCCESSFUL
ALL	DEFAULT	INTEGER	OPTION	SYNONYM
ALTER	DELETE	INTERSECT	OR	SYSDATE
AND	DESC	INTO	ORDER	TABLE
ANY	DISTINCT	IS	PCTFREE	THEN
AS	DROP	LEVEL	PRIOR	TO
ASC	ELSE	LIKE	PRIVILEGES	TRIGGER
AUDIT	EXCLUSIVE	LOCK	PUBLIC	UID
BETWEEN	EXISTS	LONG	RAW	UNION
BY	FILE	MAXEXTENTS	RENAME	UNIQUE
FROM	FLOAT	MINUS	RESOURCE	UPDATE
CHAR	FOR	MLSLABEL	REVOKE	USER
CHECK	SHARE	MODE	ROW	VALIDATE
CLUSTER	GRANT	MODIFY	ROWID	VALUES
COLUMN	GROUP	NOAUDIT	ROWNUM	VARCHAR
COMMENT	HAVING	NOCOMPRESS	ROWS	VARCHAR2
COMPRESS	IDENTIFIED	NOWAIT	SELECT	VIEW
CONNECT	IMMEDIATE	NULL	SESSION	WHENEVER
CREATE	IN	NUMBER	SET	WHERE
CURRENT	INCREMENT	OF	SIZE	WITH
DATE	INDEX	OFFLINE	SMALLINT	
CHAR	VARHCAR	VARCHAR2	NUMBER	DATE LONG
CLOB	BLOB	BFILE		
INTEGER	DECIMAL			
SUM	COUNT	GROUPING	AVERAGE	
TYPE				

● Oracle 名词解释

a) PK：Primary Key，主键
b) UK：Unique Key，唯一性索引
c) FK：Foreign Key，外键
d) DBLINK：Database Link，数据库链接

➢ Mysql

● 数据库整体设计规范（必读）

1．设计

1. 一般都使用 INNODB 存储引擎，除非读写比率<1%,才考虑使用 MYISAM 存储引擎；其他存储引擎请在 DBA 的建议下使用。
2. Stored procedure （包括存储过程，函数，触发器）对于 MYSQL 来说还不是很成熟，没有完善的出错记录处理，不建议使用。
3. UUID()，USER()这样的 MYSQL INSIDE 函数对于复制来说是很危险的，会导致主备数据.不一致。所以请不要使用。如果一定要使用 UUID 作为主键，让应用程序来产生。
4. 请不要使用外键约束，如果数据存在外键关系，请在程序层面实现。
5. 如果应用使用的是长连接，应用必须具有自动重连的机制。但请避免每执行一个 SQL 去检查一次 DB 可用性。
6. 如果应用使用的是长连接，应用应该具有连接的 TIMEOUT 检查机制，及时回收长时间没有使用的连接。 TIMEOUT 时间一般建议为 20min。
7. 我们所有的 MySQL 数据库除历史原因外，都必须采用 UTF8 编码。
8. Mysql 对 DDL 支持很差，表结构推荐设计为 Key-Value 结构。如果是关系型结构的数据库，请尽量预留一些字段，如 value1 ,value2 ,value3。
9. Mysql 用户名与数据库名字一样。

2．命名

a) 命名应使用富有意义的英文词汇，多个单词组成的，中间以下划线分割。
b) 命名只能使用英文字母，数字和下划线。
c) 命名避免使用 Mysql 的保留字（详见附录 A）和系统关键字。
d) 命名长度以不超过 15 个字符为宜(避免超过 20)。
e) 命名全部采用小写，并且名称前后不能加引号。

● 数据库对象设计规范

1. 表

设计

a) 在设计时尽量包含两个日期字段:gmt_created(创建日期),gmt_modified(修改日期)且非空, 对表的记录进行更新的时候，必须包含对 gmt_modified 字段的更新。
b) 必须要有主键，主键尽量用自增字段类型，推荐类型为 INT 或者 BIGINT 类型。
c) 需要多表 join 的字段，数据类型保持绝对一致。
d) Mysql 的表尽量设置成 KV（Key-Value）结构，这样便于扩展和维护。
e) 当表的字段数非常多时，可以将表分成两张表，一张作为条件查询表，一张作为详细内容表（主要是为了性能考虑）。
f) 当字段的类型为枚举型或布尔型时，建议使用 char(1)类型。
g) 同一表中，所有 varchar 字段的长度加起来，不能大于 65535.如果有这样的需求，请使用 TEXT/LONGTEXT 类型。
h) 由于 MYSQL 表 DDL 维护成本很高，所以在适当的时候，可以有一定的字段容余。

比如：Value1,Value2,Value3 这样的字段。

命名

a) 同一个模块的表尽可能使用相同的前缀，表名尽可能表达含义，例如：CRM_SAL_FUND_ITEM。
b) 字段命名应尽可能使用表达实际含义的英文单词或缩写，

如，公司 ID，不要使用：corporation_id, 而用：corp_id 即可。

c) 布尔值类型的字段命名为 is+描述。如 member 表上表示是否为 enabled 的会员的字段命名为 IsEnabled。

常用字段类型

TINYINT	1 个字节, -128 to 127 || 0 to 255
SMALLINT	2 个字节, -32768 to 32767 || 0 to 65535
MEDIUMINT	3 个字节, -8388608 to 8388607 || 0 to 16777215.
INT, INTEGER	4 个字节, -2147483648 to 2147483647 || 0 to 4294967295.
BIGINT	8 个字节, -9223372036854775808 to 92233720368547758080 to 18446744073709551615

DECIMAL(P,S)	定点数(以字符串形式存放)　　默认:P 为 10,S 为 0,最大 65 位
DATE	范围'1000-01-01'到'9999-12-31'　格式'YYYY-MM-DD'　(3 字节)
Time	范围'-838:59:59'到'838:59:59'　　格式'HH:MM:SS'　(3 字节)
DATETIME	范围 '1000-01-01 00:00:00' 到 '9999-12-31 23:59:59' 格式 'YYYY-MM-DD HH:MM:SS'　(8 字节)
TIMESTAMP	范围 '1970-01-01 00:00:00' 到 2037 年 格式'YYYY-MM-DD HH:MM:SS'　宽度固定为 19 个字符(4 字节) 不建议使用。
VARCHAR(n)	变长字符串，65532>n>4, 注意，n 是字符数，而不是字节数
CHAR(n)	定长字符串,n 范围(0,255)， 如果不是定长的数据，n<=4 时才使用
TINYBLOB, TINYTEXT	存储 L+1 个字节，其中 L < 2^8
BLOB, TEXT	存储 L+2 个字节，其中 L < 2^16
MEDIUMBLOB, MEDIUMTEXT	存储 L+3 个字节，其中 L < 2^24
LONGBLOB, LONGTEXT	存储 L+4 个字节，其中 L < 2^32

字段注释

a)　标准字段注释由一组"@"开头的标签+空格+文本组成。

以 MD_USER 表的部分字段为例：

Name	Type	Comments
PARTY_ID	VARCHAR(20)	@desc 主键 ID
CORP_ID	VARCHAR(20)	@desc 用户所在公司 ID @fk md_corp_id

STATUS	VARCHAR(20)	@desc 状态 @values disable|enable: 未激活状态|激活状态
IS_PRI_ACCOUNT	CHAR(1)	@desc 是否为主账号。后台生成 UK 时使用 @values y|n:是帐号，非主帐号 @logic 一个公司内部，有且仅有一个主账号存在

b) 注释标签说明

标签名	中文含义	必填	备注
@desc	字段中文描述	Yes	
@fk	字段对应的外键字段		
@values	取值范围说明。多个值以"|"分隔		如此字段的值由系统自动生成，可忽略不书写。
@sample	数据范本		对于复杂数据格式，最好给一个数据范本。
@formula	计算公式		写明该字段由哪些字段以何种公式计算得到。
@logic	数据逻辑		简要写明该字段的数据是在何种业务规则下，如何变化的。
@redu	标识此字段冗余		
@depr	标识此字段已废弃		简要写明：废弃人 废弃日期 废弃原因

2. 索引

设计

a) Bitmap 索引通常不适合我们的环境。
b) 索引根据实际 SQL，由 DBA 创建。
c) 不要创建带约束的索引，所有的约束效果都通过显示创建约束然后再 using index 一个已经创建好的普通索引来实现。

命名

a) <table_name>_<column_name>_ind,各部分以下划线（_）分割。
b) 多单词组成的 column name，取前几个单词首字母，加末单词组成 column_name。如 sample 表 member_id 上的索引：sample_mid_ind。

3. 约束

设计

a) 主键最好是无意义的，，统一由 Auto-Increment 字段生成整型数据，不建议使用组合主键。
b) 若要达到唯一性限制的效果，不要创建 unique index，必须显式创建普通索引和约束（pk 或 uk），即先创建一个以约束名命名的普通索引，然后创建一个约束，用 using index ...指定索引。
c) 当删除约束的时候，为了确保不影响到 index，最好加上 keep index 参数。
d) 主键的内容不能被修改。
e) 外键约束一般不在数据库上创建，只表达一个逻辑的概念，由程序控制。
f) 当万不得已必须使用外健的话，必须在外健列创建 INDEX。

命名

a) 主键约束： _pk 结尾，<table_name>_pk；
b) unique 约束：_uk 结尾，<table_name>_<column_name>_uk；
c) check 约束： _ck 结尾，<table_name>_<column_name>_ck；
d) 外键约束： _fk 结尾，以 pri 连接本表与主表，<table_name>_pri_<table_name>_fk；

4. 触发器

命名

a) <table_name>_A(After)B(Before)I(Insert)U(Update)D(Delete)_trg。
b) 若是用于同步的触发器以 sync 作为前缀：sync_<table_name>_trg。

5. 过程、函数

设计

a) 如果要在 MYSQL 里使用存储过程类的技术，请务必和 DBA 沟通确认。

命名

a) 过程以 proc_开头，函数以 func_开头。
b) 变量命名约定：本地变量以 v_为前缀，参数以 p_为前缀，可以带_I(输入)，_O(输出)、_IO(输入输出）表示参数的输入输出类型。

● SQL 开发规范

一. 编码规范

1. 使用 SQL 操作数据库前，必须由 use DB_name 开始

```
Use Test ;
Insert into Table_name values ( … ) ;
Commit;
```

2. 如果需要事务的支持，在确认使用了 innodb 存储引擎的前提下，在数据库连接时，先关闭自动提交

比如，设定 set auto_commit =0 ;

3. 写到应用程序里的 SQL 语句，禁止一切 DDL 操作

例： Create table ， Drop table ， Create database ， Drop database ，
Alter table ，grant … …
如有特殊需要，必需与 DBA 协商同意方可使用。

4. 获取当前时间请使用 now()，不要用 sysdate()来代替

这对复制来说是很危险的，会导致主从数据不一致的情况；
因为 sysdate,取的是系统主机时间，在 BINLOG 会原文传输，
当在应用时会与主库产生差异。

5. 写 SQL 的时候一定要给每个字段指定表名做前缀

比如:

```
select a.id,a.name from test a;
```

好处是 一来带来性能的提升，
二来可以避免一些错误的发生。

6. 在 iBatis 的 SqlMap 文件中绑定变量使用 “#var_name#”表示,替代变量使用“var_name”

所有需要动态 Order By 条件的 Query，在使用替代变量过程中，需要将可能传入的内容以枚举类写死在代码中，禁止接受任何外部传入内容。

7. 请不要写 select * 这样的代码，指定需要的字段名

8. Mysql 对日期（datetime）允许“不严格”语法

任何标点符都可以用做日期部分或时间部分之间的间割符。
例如，
'98-12-31 11:30:45'、'98.12.31 11+30+45'、'98/12/31 11*30*45'和'98@12@31 11^30^45'是等价的。]
我们自己约定一种写法，与 Oracle 相通： '2009-12-31 11:30:45'

9. Mysql 的日期与字符是相同的，所以不需要做另外的转换

例：

```
Select e.username from employee e where
        e.birthday >=’1998-12-31 11:30:45’
```

10. 避免多余的排序。使用 GROUP BY 时，默认会进行排序，当你不需要排序时，可以使用 order by null

```
Select product,count(*) cnt from crm_sale_detail group by product order by null;
```

11. 避免在 where 子句中对字段施加函数

a) 通常，不允许在字段上添加函数或者表达式，这样将导致索引失效，如：
错误的写法：

```
select * from iw_account_log where substr(username,1,5)=’abcde’
```

正确的写法：

```
select * from iw_account_log where username like ’abcde%’
```

b) 如果是业务要求的除外，但需要在编写时咨询 DBA
c) 特别注意，当表连接时，用于连接的两个表的字段如果数据类型不一致，则必须在一边加上类型转换的函数

12. 严格要求使用正确类型的变量，杜绝 Mysql 做隐式类型转换的情况

13. 全模糊查询无法使用 INDEX，应当尽可能避免

比如：select * from table where name like '%jacky%';

14. 表连接规范

a) 所有非外连接 SQL（即 INNER JOIN），请把关联表统一写到 FROM 字句中，关联条件与过滤条件统一写到 WHERE 字句中
b) 出于代码的可读性原因，所有外连接 SQL 语句中，请一律使用 LEFT JOIN，禁用 RIGHT JOIN
c) 另外，请注意 LEFT JOIN 字句中，右边位置表的条件书写位置不同的影响：

```
SELECT A.rolename,A.gmt_create,B.nickname FROM gl_role A LEFT JOIN
```

```
        gl_roledetail B ON A.ID=B.roleid AND B.roleID=2;
+-------------+---------------------+----------+
| rolename    | gmt_create          | nickname |
+-------------+---------------------+----------+
| 163.com     | 0000-00-00 00:00:00 | test2    |
| sina.com    | 0000-00-00 00:00:00 | NULL     |
| hotmail.com | 0000-00-00 00:00:00 | NULL     |
| 126.com     | 2009-08-20 18:20:18 | NULL     |
+-------------+---------------------+----------+
```

```
SELECT A.rolename,A.gmt_create,B.nickname FROM gl_role A LEFT JOIN
        gl_roledetail B ON A.ID=B.roleid WHERE B.roleID=2;
+----------+---------------------+----------+
| rolename | gmt_create          | nickname |
+----------+---------------------+----------+
| 163.com  | 0000-00-00 00:00:00 | test2    |
+----------+---------------------+----------+
```

15. 表连接分页查询的使用

a) 常规分页语句写法(start：起始记录数，page_offset：每页记录数)：
SELECT ID,username FROM gl_user WHERE username like '%@163.com' ORDER BY M.gmt_create LIMIT start, page_offset;

b) 多表 Join 的分页语句，如果过滤条件在单个表上，需要先分页，再 Join：
低性能写法：
SELECT M.username,P.rolename FROM gl_user M INNER JOIN gl_role P ON M.ID=P.userid WHERE username like '%@163.com' ORDER BY M.gmt_create LIMIT start, page_offset;
高性能写法：
SELECT M.username,P.rolename
FROM (SELECT ID,username FROM gl_user WHERE username like '%@163.com' ORDER BY M.gmt_create LIMIT start, page_offset)M,gl_role P
WHERE M.ID=P.userid;
这样写的前提是关联的表之间记录一一对应，否则可能会返回的记录数目少于或多余 page_offset 的值。

16. "join"、"in"、"not in"、"exsits"和"not exists"的使用

a) 比较 IN,EXISTS,JOIN
按效率从好到差排序:
字段上有索引 : EXISTS, IN, JOIN
字段上没有索引: JOIN, EXISTS ,IN

b) Anti-Joins: NOT IN ,NOT EXISTS, LEFT JOIN

按效率从好到差排序:
字段上有索引 : LEFT JOIN, NOT EXISTS, NOT IN
字段上没有索引: NOT IN, NOT EXISTS, LEFT JOIN

17. 其它编写规范

a) 对表的记录进行更新的时候，必须包含对 gmt_modified 字段的更新;
b) 不允许在 where 后添加 1=1 这样的无用条件，where 可以写在 prepend 属性里，如：

错误的写法：

```
select count      from BD_CONTRACT t where 1=1
<dynamic>
  ......
</dynamic>
```

正确的写法：

```
select count      from BD_CONTRACT t
<dynamic prepend="where">
  ......
</dynamic>
```

c) 对大表进行查询时，在 SQLMAP 中需要加上对空条件的判断语句，具体可在遇到时咨询 DBA，如：

性能上不保险的写法：

```
select count      from iw_user usr
<dynamic prepend="where">
  <isNotEmpty prepend="AND" property="userId">
    usr.iw_user_id = #userId:varchar#
  </isNotEmpty>
  <isNotEmpty prepend="AND" property="email">
    usr.email = #email:varchar#
  </isNotEmpty>
  <isNotEmpty prepend="AND" property="certType">
    usr.cert_type = #certType:varchar#
  </isNotEmpty>
  <isNotEmpty prepend="AND" property="certNo">
    usr.cert_no = #certNo:varchar#
  </isNotEmpty>
</dynamic>
```

性能上较保险的写法（防止那些能保证查询性能的关键条件都为空）：

```
select count      from iw_user usr
<dynamic prepend="where">
 <isNotEmpty prepend="AND" property="userId">
  usr.iw_user_id = #userId:varchar#
 </isNotEmpty>
 <isNotEmpty prepend="AND" property="email">
  usr.email = #email:varchar#
 </isNotEmpty>
 <isNotEmpty prepend="AND" property="certType">
  usr.cert_type = #certType:varchar#
 </isNotEmpty>
 <isNotEmpty prepend="AND" property="certNo">
  usr.cert_no = #certNo:varchar#
 </isNotEmpty>
 <isEmpty property="userId">
  <isEmpty property="email">
   <isEmpty property="certNo">
    query not allowed
   </isEmpty>
  </isEmpty>
 </isEmpty>
</dynamic>
```

另外，对查询表单的查询控制建议使用 web 层进行控制而不是客户端脚本（JAVASCRIPT/VBSCRIPT）

d) 聚合函数常见问题

1) 不要使用 count(1)代替 count（*）

2) count(column_name)计算该列不为 NULL 的记录条数

3) count(distinct column_name)计算该列不为 NULL 的不重复值数量

4) count()函数不会返回 NULL，但 sum()函数可能返回 NULL，可以使用 ifnull(sum(qty),0)来避免返回 NULL

e) NULL 的使用

1) 理解 NULL 的含义，是"不确定"，而不是"空"

2) 查询时，使用 is null 或者 is not null

3) 更新时，使用等于号，如：update tablename set column_name = null

二．格式规范

1．注释说明

a) 本注释说明主要用于 Mysql Client 程序及其它 SQL 文件，其它可作参考；
b) SQL 接受的注释有三种：

```
-- 这儿是注释 (注意，第 2 个破折号后面至少跟一个空格符)
/* 这儿是注释 */
# 这儿是注释
```

c) 下面的例子显示了 3 种风格的注释：

```
mysql> SELECT 1+1;     # This comment continues to the end of line
mysql> SELECT 1+1;     -- This comment continues to the end of line
mysql> SELECT 1 /* this is an in-line comment */ + 1;
mysql> SELECT 1+
/*
this is a
multiple-line comment
*/
```

2．缩进

低级别语句在高级别语句后的，一般缩进 4 个空格：

```
DECLARE
    v_MemberId      VARCHAR(32),
BEGIN
    SELECT admin_member_id INTO v_MemberId
      FROM company
      WHERE id = 10;
    SELECT v_MemberId ;
END;
```

同一语句不同部分的缩进，如果为 sub statement，则通常为 2 个空格，如果与上一句某部分有密切联系的，则缩至与其对齐：

```
BEGIN
FOR v_TmpRec IN
(SELECT login_id,
                    gmt_created, -- here indented as column above
                      satus
                FROM  member            -- sub statement
                WHERE site = 'china'
                    AND country='cn' )
    LOOP
        NULL;
    END  LOOP;
END;
```

3．断行

a) 一行最长不能超过 80 字符
b) 同一语句不同字句之间
c) 逗号以后空格
d) 其他分割符前空格

```
SELECT  concat(offer_name,',',
      offer_count as offer_category,
       id)
 FROM category
 WHERE super_category_id_1 = 0;
```

- 附录：Mysql 保留字

ADD	DEFAULT	INSERT	NULL	SQL_CALC_FOUND_ROWS
ALL	DELAYED	INT	NUMERIC	SQL_SMALL_RESULT
ALTER	DELETE	INT1	ON	SQLEXCEPTION
ANALYZE	DESC	INT2	OPTIMIZE	SQLSTATE
AND	DESCRIBE	INT3	OPTION	SQLWARNING
AS	DETERMINISTIC	INT4	OPTIONALLY	SSL
ASC	DISTINCT	INT8	OR	STARTING
ASENSITIVE	DISTINCTROW	INTEGER	ORDER	STRAIGHT_JOIN
BEFORE	DIV	INTERVAL	OUT	TABLE
BETWEEN	DOUBLE	INTO	OUTER	TERMINATED
BIGINT	DROP	IS	OUTFILE	THEN
BINARY	DUAL	ITERATE	PRECISION	TINYBLOB
BLOB	EACH	JOIN	PRIMARY	TINYINT
BOTH	ELSE	KEY	PROCEDURE	TINYTEXT
BY	ELSEIF	KEYS	PURGE	TO
CALL	ENCLOSED	KILL	RAID0	TRAILING
CASCADE	ESCAPED	LABEL	RANGE	TRIGGER
CASE	EXISTS	LEADING	READ	UNDO
CHANGE	EXIT	LEAVE	READS	UNION
CHAR	EXPLAIN	LEFT	REAL	UNIQUE
CHARACTER	FETCH	LIKE	REFERENCES	UNLOCK
CHECK	FLOAT	LIMIT	REGEXP	UNSIGNED
COLLATE	FLOAT4	LINEAR	RELEASE	UPDATE
COLUMN	FLOAT8	LINES	RENAME	USAGE
CONDITION	FOR	LOAD	REPEAT	USE
CONNECTION	FORCE	LOCALTIME	REPLACE	USING
CONSTRAINT	FOREIGN	LOCALTIMESTAMP	REQUIRE	UTC_DATE
CONTINUE	FROM	LOCK	RESTRICT	UTC_TIME
CONVERT	FULLTEXT	LONG	RETURN	UTC_TIMESTAMP
CREATE	GOTO	LONGBLOB	REVOKE	VALUES
CROSS	GRANT	LONGTEXT	RIGHT	VARBINARY
CURRENT_DATE	GROUP	LOOP	RLIKE	VARCHAR
CURRENT_TIME	HAVING	LOW_PRIORITY	SCHEMA	VARCHARACTER
CURRENT_TIMESTAMP	HIGH_PRIORITY	MATCH	SCHEMAS	VARYING
CURRENT_USER	HOUR_MICROSECOND	MEDIUMBLOB	SECOND_MICROSECOND	WHEN
CURSOR	HOUR_MINUTE	MEDIUMINT	SELECT	WHERE
DATABASE	HOUR_SECOND	MEDIUMTEXT	SENSITIVE	WHILE

DATABASES	IF	MIDDLEINT	SEPARATOR	WITH
DAY_HOUR	IGNORE	MINUTE_MICROSECOND	SET	WRITE
DAY_MICROSECOND	IN	MINUTE_SECOND	SHOW	X509
DAY_MINUTE	INDEX	MOD	SMALLINT	XOR
DAY_SECOND	INFILE	MODIFIES	SPATIAL	YEAR_MONTH
DEC	INNER	NATURAL	SPECIFIC	ZEROFILL
DECIMAL	INOUT	NO_WRITE_TO_BINLOG	SQL	FALSE
DECLARE	INSENSITIVE	NOT	SQL_BIG_RESULT	TRUE

- Mysql 名词解释

a) PK：Primary Key，主键
b) UK：Unique Key，唯一性索引
c) FK：Foreign Key，外键