タイトル:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者・会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者・会社略称	史 荣新 CJB	新規作成者・会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

情 報 種 別：社外秘
会　社　名：長春必捷必信息技術有限公司
情報所有者：開発部

社内共通規約

SQL規約篇

長春必捷必(CJB)信息技術有限公司
開発部

タイトル:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者・会社略称	喬 恒斌　CJB	最終更新日	2016.04.18
		1.00	最終承認者・会社略称	史 荣新　CJB	新規作成者・会社略称	喬 恒斌　CJB	新規作成日	2016.04.18

タイトル:	社内共通規約−SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者・会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者・会社略称	史 荣新 CJB	新規作成者・会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

変更履歴

項番	変更種別	変更内容	変更者	変更日	承認者	承認日
1	新規作成	初回作成	喬　恒斌	4月18日	史 荣新	4月19日
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

タイトル:	社内共通規約−SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者・会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者・会社略称	史 荣新 CJB	新規作成者・会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

29						
30						

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

目次

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

ﾀｲﾄﾙ:	社内共通規約−SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

本规约书的前提

本规约的适用于所有开发环境的DB为ORACLE，MICROSOFT SQL SERVER或者DB2的JAVA、MICROSOFT .NET产品的开发作业，
本规约作为<必须要遵守的、最底限的编码规约>以及<在开发和设计层面的、最底限的考虑点>的等级需要各开发团队遵守。

本规约书的目的

通过本共通的标准化规约，使<长春必捷必信息技术有限公司>(以下简称必捷必)负责的产品达到效果是本规约的根本目的。

1．SQL性能的改善(高效的数据访问)
2．SQL相关典型问题的有效避免（死锁等）
3．品质稳定化，提高开发效率
4．统一的SQL编写方式带来的系统易维护

※1　本规约未明确记载的SQL在原则上不允许使用。但是，如果不使用本规约未明确的SQL会导致业务实现困难的场合，要对个别业务进行
　　个别精查的基础上，进行否需要使用本规约之外的SQL的判断（需得到PM以上管理人员的同意）

ﾀｲﾄﾙ:	社内共通規約−SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌　CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新　CJB	新規作成者･会社略称	喬 恒斌　CJB	新規作成日	2016.04.18

※2　BATCH案件中利用的SQL，不受本规约限制。

ﾀｲﾄﾙ:	社内共通規約−SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

关于本规约中使用的示例

本规约中使用的示例通常将表命名为TBLXXX，将字段(列)命名为COLXXX，除此之外，本规约也会使用EMP(雇员表)，部门表（DEPT）、工程表（Project）这3各具体的数据表

EMP

名称	数据类型	长度	NULL
EMPNO	CHARACTER	6	不可
FIRSTNME	VARCHAR	12	不可
MIDINIT	CHARACTER	1	可
LASTNAME	VARCHAR	15	不可
WORKDEPT	CHARACTER	3	可
PHONENO	CHARACTER	4	可
HIREDATE	DATE	4	可
JOB	CHARACTER	8	可
EDLEVEL	SMALLINT	2	buke
SEX	CHARACTER	1	可
BIRTHDATE	DATE	4	可
SALARY	DECIMAL	9	可
BONUS	DECIMAL	9	可
COMM	DECIMAL	9	可

DEPT

名称	数据类型	长度	NULL
DEPTNO	CHARACTER	3	不可
DEPTNAME	VARCHAR	36	不可
MRGNO	CHARACTER	6	可
ADMRDEPT	CHARACTER	3	不可
LOCATION	CHARACTER	16	可

PROJECT

名称	数据类型	长度	NULL
PROJNO	CHARACTER	6	不可
PROJNAME	VARCHAR	24	不可
DEPTNO	CHARACTER	3	不可
RESPEMP	CHARACTER	6	不可
PRSTAFF	DECIMAL	5	可
PRSTDATE	DATE	4	可
PREDDATE	DATE	4	可
MAJPROJ	CHARACTER	6	可

ﾀｲﾄﾙ:	社内共通規約−SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌　CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新　CJB	新規作成者･会社略称	喬 恒斌　CJB	新規作成日	2016.04.18

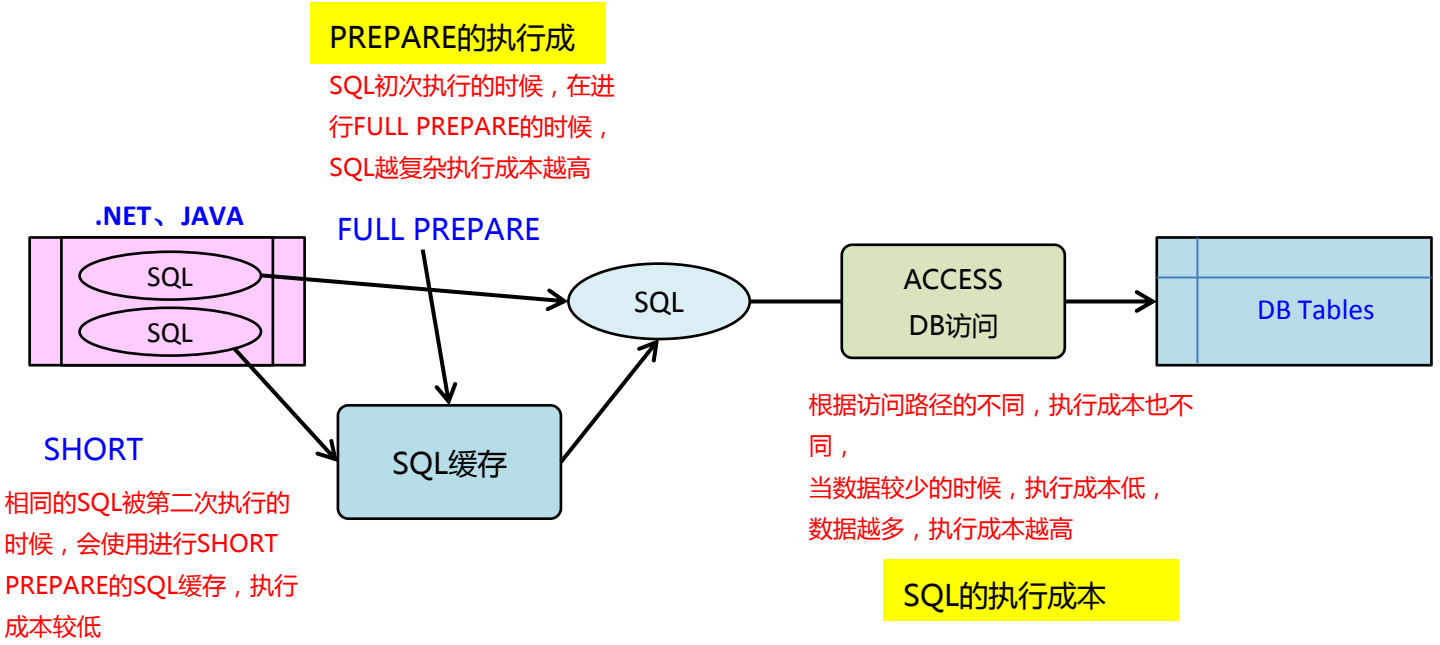
2 SQL的编写

2-1　编写简单的SQL

1）　SQL逻辑尽可能简单，另令DB服务器的处理负荷尽可能低

2）　理由

使用动态SQL的时候，复杂的SQL的PAREPARE成本会变高
SQL简单的话，PREPARE处理的成本会有效减少
DB服务器可能和其他系统构建在一台物理服务器上，发行高负荷的SQL会影响其他系统的正常运行，
因此要设计可以低负荷执行的SQL。（即使SQL简单，如果查询条件不慎重设计的话执行负荷也会变的很高）
本规约在不引入过分复杂繁琐的规约的前提下，为实现以上目的对SQL编写规则进行了最低限的定义



2-2　保证SQL的可读性

タイトル:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

1）　统一记述内容

･使用文字

表名，列名等定义文使用和DB一样的大小写，别名的大小写要和DB表字段的命名方式相同

SQL关键字一律使用大文字

･空白

SQL文中的空白，要满足如下要求

･换行前的SQL文末尾不要有空白

･空白使用一个半角空格，不能存在连续3个以上的空格（SQL缩进除外）

･缩进

使SQL保持良好的可读性，在适当的地方进行缩进，缩进的单位原则上为4个半角空格

･换行

对一下的位置进行换行

<SELECT文>

首个项目（字段，表名）之前
相同种类的项目的时候，逗号（，）之后
FROM的前后
WHERE的前后
AND，OR的前面
GROUP BY的前后
HAVING的前后
ORDER BY的前后
UNION ALL的前后
INNER JOIN的前面
LEFT OUT JOIN的前面

示例

```
SELECT
   PROJNO,
   DEPTNO,
   PRSTDATE,
   PRENDATE,
FROM
   PROJECT
WHERE
   DEPTNO >= 50
ORDER BY
   PROJNO
```

<INSERT文>

登录对象列的括号的后面
同类项目（列等）间分隔符（逗号）后面
VALUES的前面

示例

```
INSERT INTO EMP(
   EMP_CD,
   ENAME)
VALUES(
```

タイトル:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

```
    9999,
    'ITO';)
```

<UPDATE文>
　SET的前后
　同类项目（列等）间分隔符（逗号）后面
　WHERE的前后
　AND，OR的前面

示例

```
UPDATE EMP(
SET
   ENAME =   'ITO'
WHERE
   EMP_CD = 9999)
```

<UPDATE文>
　DELETE的后面
　WHERE的前后
　AND，OR的前面

示例

```
DELETE
FROM EMP
WHERE
   EMP_CD = 9999
```

· 表别名的利用方法
　表别名的命名原则T#，#为从1开始的连续整数，顺序为自顶向下。

· SQL文最大长度
　SQL文最大长度不能超过32K，如果超过32K要进行拆分
　DB2　　：HOST DB2 最大长度为32K，V8版本以上为2MB字节
　ORACLE：对于SQL文，11G以前为32K，11G以后无特别限制，对于PL/SQL，UNIX上最大是64K，WINDOWS则是32K
　MS SQL：65,536 * 网络数据包大小

2）　理由
　团队成员编写的SQL文不会各式各样，进而提高品质

3）　特别事项

受制与本资料的页面布局限制，示例代码可能会有和规约不一样的地方

2-3 否定形式的编写方式

1） 详细内容

不等于号不要使用某DB的专有运算符(!=、~=、^=等)，要使用各DB通用的运算符

不等于用　　<>

大于等于用　>=

小于等于 用　<=

2） 理由

如果使用个别DBMS专有的符号，当系统将来需要变更为其他DBMS的时候，有发生异常的可能

2-4 日期项目的判定方式

1） 详细内容

CHAR类型的日期项目，对其的值存在的检查不可使用NOT条件，要使用 日期 > '00000000'

2） 理由

日期 <> ' '(半角空白) 这样的指定条件虽然可以执行并得到正确结果，但是为了对NOT条件进行判定，SQL引擎会进行全数据读取，另外，如果使用NOT条件，也无法有效利用索引

3） 特别事项

本规约的适用前提是“该CHAR类型日期项目中的数据一定会是空格或者全数字 “。

2-5 SQL的相关限制

1） 详细内容

项目	限制
表名	最大30个半角字符，以字母开头，可包含字母和数字，以及下划线
索引名	最大30个半角字符，以字母开头，可包含字母和数字，以及下划线
View名	最大30个半角字符，以字母开头，可包含字母和数字，以及下划线
字段名	最大30个半角字符，以字母开头，可包含字母和数字，以及下划线

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

3 SQL基本规约

3-1 查询SQL中不可使用 SELECT *

1） 详细内容

对于SELECT文，原则上只能对必要的列进行查询
但如果从易维护的观点，进行SQL共通化处理的化，不受此原则限制。
如果需要取得全列，不能使用SELECT *，而是要把所有列一一明记

2） 理由

(1) 只查询必要的列

· 可以有效削减临时存储空间
防止因为检索非必要的无用列而是临时存储空间被浪费
· 高效的数据访问
只查询必要的列，可以提高索引使用的命中率，使查询处理更加高效
· 降低系统开销
抑止DB和软件系统之间无用的数据交互

(2) 不使用 *

当表的列进行追加/变更/删除的时候，对软件系统的修改规模会最大程度的减少

3） 特别事项

当所开发模块是用于维护，备份系统的相关模块的时候，为了当DB式样发生变更时尽量减少影响，
则推荐使用SELECT *

3-2 集约函数的有效利用

1） 详细内容

可以使用的集约函数限制在合计（SUM），最大（MAX），最小（MIN），平均（AVG），对象数（COUNT）5个函数
当业务里有合计（SUM），最大（MAX），最小（MIN），平均（AVG），对象数（COUNT）的处理的时候，不允许
在DB取得基本数据后，在应用程序中进行计算的实现方式，要使用对应的SQL 集约函数

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

例）
获取所有雇员的SALARY的平均值

SELECT SALARY FROM EMPLOYEE 进行查询，然后在应用程序中进行计算
↓
SELECT AVG(SALARY) FROM EMPLOYEE

2）　理由
可以有效避免以下的缺点
- 应用程序的逻辑追加，工数增长
- 应用程序的逻辑上发生BUG
- 传递给应用程序的数据量的增加

3）　特别事项
使用集约函数的时候，要注意以下的3-3 、3-4、 3-5

3-3　当不存在指定数据的场合使用集约函数的注意点

1）　详细内容
在SELECT语句里使用MAX、MIN、SUM、AVG、COUNT等集约函数的时候，即使操作的对象字段是[NOT NULL]属性，
当数据不存在的时候，MAX、MIN、SUM、AVG会返回NULL，COUNT会返回0
因此，MAX、MIN、SUM、AVG的结果集（AnswerSet）中，要将相应的字段设置为[NULL]属性

此外，当数据不存在的时候，SELECT语句执行后的SQLCODE的值也需要注意，不实用集约函数的SQL返回的是100（Not Found），
使用集约函数的SQL返回的是0（正常终了）

2）　理由
通常情况下，判断SELECT SQL的执行的结果有无，是通过SQLCODE=100来实现的，但是当使用集约函数的场合，对SQLCODE
的使用要额外注意（使用集约函数的SQL，即使没有数据也会返回0）

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

3） 特別事項

无特別事項

3-4 关于COUNT（*）的使用

1） 详细内容

由于在没有WHERE条件的时候，使用COUNT(*)会自动调用唯一性索引（Unique Index），所以尽量使用COUNT(*)

2） 理由

如果COUNT(字段)，当字段为[NULL]属性的时候，COUNT函数会无视该行记录，当字段为[NOT NULL]的时候，
才会得到正确的结果，而COUNT(1)执行的时候并不会去调用索引，因此性能也会比COUNT(*)差

3） 特別事項

无特別事項

3-5 若只是为确认是否存在数据，不可使用COUNT（*）仅针对DB2

1） 详细内容

若只是要确认对象数据是否存在（1件以上），不允许使用COUNT(*)

例）

检查是否有薪水达到40000以上的雇员

```
SELECT COUNT(*) FROM EMPLOYEE WHERE SALARY >= 4000
↓
SELECT SALARY FROM EMPLOYEE WHERE SALARY >= 4000 FETCH FIRST 1 ROWS ONL\DB2
```

2） 理由

可降低CPU开销，防止锁冲突，减少处理时间

3） 特別事項

无特別事項

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌　CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新　CJB	新規作成者･会社略称	喬 恒斌　CJB	新規作成日	2016.04.18

3-6 关于ORDER BY，GROUP BY的注意点

1） 详细内容

使用ORDER BY的字段，升顺ASC无需明记，只有降顺需要明记

使用ORDER BY是，操作字段尽量为索引字段

原则上ORDER BY，GROUP BY的目标数据量如下

- BATCH系统，目标数据量不超过10万件
- OnLine系统，目标数据量不超过1000件

若业务上不得不超过本要求，必须认真检讨时候没有其他方法实现本业务

2） 理由

ORDER BY，GROUP BY执行时会生成临时表，如果数据过多，会导致如下问题

- 临时表的I/O处理大幅增加
- 临时表没有索引，数据扫描是全数据扫描，性能会恶化
- 单个处理大量使用，会导致其他处理出现临时表空间不住，发生异常

3） 特别事项

无特别事项

3-7 关于UNION，UNION ALL的注意点

1） 详细内容

UNION，UNION ALL，根据结果集中重复数据的有无，取得的结果会不同

UNION　　对各子查询的结果进行汇总，并去掉重复行

UNION ALL　对各子查询的结果进行汇总，不做删除重复行的处理

除非业务上必须要用UNION ，原则上只能使用UNION ALL

2） 理由

UNION会做去除重复行处理，因此会进行SORT排序，为了削减SORT处理负荷，除非业务要求，

要尽量使用UNION ALL

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

3） 特别事项

无特别事项

3-8 如何进行高效的结合(JOIN)查询

1） 详细内容

为了保持结合处理的高效执行，结合形式要满足如下要求

· 结合对象的表，要现对各个表进行业务上的过滤，只获取对象记录

· 过滤的条件原则上要使用索引列

· 结合条件原则上使用索引列

※结合处理的表数，OUTER JOIN需要注意的点等相关内容，请参照后续章节

2） 理由

对于结合处理，根据结合方式和数据量，执行效率有极度恶化的可能

对于结合处理的执行时间和CPU使用量要十分注意

为了尽可能的控制结合处理的执行代价，有效的利用索引是很重要的手段

有效利用索引，进行嵌套循环连接(Nested Loops)是结合处理优化的基本功

3） 特别事项

SQL Server支持三种物理连接,即：嵌套循环连接(Nested Loops), 合并联接(Merge), 哈希联接(Hash)

1.嵌套循环连接(Nested Loops)适用范围

两个表, 一个叫外部表, 一个叫内部表.如果外部输入非常小，而内部输入非常大并且已预先建立索引，那么嵌套循环联接将特别有效率

关于连接时哪个表为outer表，哪个为inner表，sql server会自动给你安排，和你写的位置无关，它自动选择数据量小的表为outer表， 数据量大的表为inner表。

2.合并联接(Merge)

指两个表在on的过滤条件上都有索引, 都是有序的, 这样, join时, sql server就会使用Merge join, 这样性能更好.

如果一个有索引,一个没索引,则会选择Nested Loops join.

3.哈希联接(Hash)

如果两个表在on的过滤条件上都没有索引, 则就会使用Hash join.

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

也就是说, 使用Hash join算法是由于缺少现成的索引.

参考资料:

Inside SQL server 2008 T-SQL.

高级查询优化概念

http://msdn.microsoft.com/zh-cn/library/ms191426(v=SQL.100).aspx

3-9 表结合数最大值

1） 详细内容

原则上一次SQL处理只允许最多结合6张表（数据量较小的CODE表，根据索引KEY，对应关系为1:1的场合除外），
当结合中含有VIEW的时候，VIEW中结合的表也属于计算对象
如果必须要对6个以上的表进行结合处理，必须先经过如下检讨并获得PL以上管理人员同意

· 进行去范式化，减少需要结合的表的数量
· 将SQL分成多块进行，使每块的SQL结合的表的数量减少
· 对每个表附加相对严格的过滤条件，减少各个结合表的数据量

2） 理由

即使是对3个以上的表进行结合，SQL引擎每次结合也只是对2个表进行结合，然后将这2张表的结合结果放到临时表，
然后在用此临时表结合下1张物理表，如此循环知道结合完所有的表。
当结合的表过多的时候，根据DB内部维护的表信息以及指定的结合条件，结合的顺序会发生变化，执行时间也会变化
尤其是对非唯一值的列之间进行M：N进行结合的时候，会出现大量消耗临时表空间的情景。
因此，给予以下观点，对表的结合数进行限制

· 执行成本的稳定
· 防止因为临时表膨胀导致的性能恶化
· 发生性能问题是的调校作业简单化

3） 特别事项

当结合表过多采用对应如上对应方法的时候，很可能需要对应用程序进行修改，有可能带来开发作业量的增加
虽然本规约要求最大结合表数为6，但可实际开发的时候，要根据实际的数据量，性能指标，对本数值进行修改（增大或减少）

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

3-10 禁止使用FULL OUTER JOIN

1） 详细内容

禁止使用 FULL OUTER JOIN

2） 理由

FULL OUTER JOIN有可能使临时表急剧增大

3） 特别事项

无特别事项

3-11 OUTER JOIN的写法不同导致的结果不同

1） 详细内容

对于结合条件的编写位置，要根据业务要求谨慎分析。

使用OUTER JOIN 的时候，条件写在结合部和写在WHERE部中，得到的结果是不一样的。（INNER JOIN无此情况）

例）

在ON语句和在WHERE语句指定[雇佣年月日]的条件，取得的结果集是不同的

```
$ db2 "select DEP.DEPTNO,EMP.WORKDEPT,EMP.EMPNO,EMP.FIRSTNME,EMP.HIREDATE
from EMPLOYEE as EMP
left outer join DEPARTMENT as DEP
   on EMP.WORKDEPT = DEP.DEPTNO
  and EMP.HIREDATE > '1970-01-01'
where EMP.WORKDEPT <= 'C01'
order by EMP.EMPNO"

DEPTNO WORKDEPT EMPNO  FIRSTNME     HIREDATE
------ -------- ------ ------------ ----------
-      A00      000010 CHRISTINE    1965-01-01
B01    B01      000020 MICHAEL      1973-10-10
C01    C01      000030 SALLY        1975-04-05
-      A00      000110 VINCENZO     1958-05-16
-      A00      000120 SEAN         1963-12-05
C01    C01      000130 DOLORES      1971-07-28
C01    C01      000140 HEATHER      1976-12-15

  7 レコードが選択されました。
```

```
$ db2 "select DEP.DEPTNO,EMP.WORKDEPT,EMP.EMPNO,EMP.FIRSTNME,EMP.HIREDATE
from EMPLOYEE as EMP
left outer join DEPARTMENT as DEP
   on EMP.WORKDEPT = DEP.DEPTNO
where EMP.HIREDATE > '1970-01-01'
  and EMP.WORKDEPT <= 'C01'
order by EMP.EMPNO"

DEPTNO WORKDEPT EMPNO  FIRSTNME     HIREDATE
------ -------- ------ ------------ ----------
B01    B01      000020 MICHAEL      1973-10-10
C01    C01      000030 SALLY        1975-04-05
C01    C01      000130 DOLORES      1971-07-28
C01    C01      000140 HEATHER      1976-12-15

  4 レコードが選択されました。
```

タイトル:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

在ON语句中指定　　*在WHERE语句中指定*

2） 理由

[条件匹配]和[执行结合]的顺序变化，必然会导致结果集的变化，因此要谨慎分析业务，选择正确的编写方式

3） 特别事项

若使用的是DB2的V7版本，为了将中间表空间最小化，上面WHERE语句中的条件，应该提到结合中采用嵌套查询的形式

3-12 OUTER JOIN处理中关于NULL值的回避

1） 详细内容

当OUTER JOIN处理中没有匹配的数据的时候，即使某个字段在DB中是[NOT NULL],取得的结果也会是NULL，
如果业务上在返回NULL的时候会发生问题，就要对结果集的相应字段使用COALESCE函数，将NULL变成有效的值

例）

查询部门经理的部门编号(DEPTNO)和薪水(SALARY)的时候，D01部门尚没有部门经理

不使用COALESCE函数

```
$ db2 "select empno,deptno,firstnme,deptname,salary from department as
dep left outer join employee as emp on dep.mgrno=emp.empno"

EMPNO  DEPTNO FIRSTNME     DEPTNAME                       SALARY
------ ------ ------------ ------------------------------ -----------
000010 A00    CHRISTINE    SPIFFY COMPUTER SERVICE DIV.     52750.00
000020 B01    MICHAEL      PLANNING                         41250.00
000030 C01    SALLY        INFORMATION CENTER               38250.00
-      D01    -            DEVELOPMENT CENTER                   -
000060 D11    IRVING       MANUFACTURING SYSTEMS            32250.00
000070 D21    EVA          ADMINISTRATION SYSTEMS           36170.00
000050 E01    JOHN         SUPPORT SERVICES                 40175.00
000090 E11    EILEEN       OPERATIONS                       29750.00
000100 E21    THEODORE     SOFTWARE SUPPORT                 26150.00

  9 レコードが選択されました。
```

使用COALESCE函数

```
$ db2 "select coalesce(empno,'n/a'),deptno,firstnme,deptname,coalesce(salary,0)
from department as dep left outer join employee as emp on dep.mgrno=emp.empno"

1      DEPTNO FIRSTNME     DEPTNAME                       5
------ ------ ------------ ------------------------------ --------------
000010 A00    CHRISTINE    SPIFFY COMPUTER SERVICE DIV.         52750.00
000020 B01    MICHAEL      PLANNING                             41250.00
000030 C01    SALLY        INFORMATION CENTER                   38250.00
n/a    D01    -            DEVELOPMENT CENTER                       0.00
000060 D11    IRVING       MANUFACTURING SYSTEMS                32250.00
000070 D21    EVA          ADMINISTRATION SYSTEMS               36170.00
000050 E01    JOHN         SUPPORT SERVICES                     40175.00
000090 E11    EILEEN       OPERATIONS                           29750.00
000100 E21    THEODORE     SOFTWARE SUPPORT                     26150.00

  9 レコードが選択されました。
```

2） 理由

无特别理由

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

3） 特別事項

无特别事项

3-13 禁止对VIEW进行更新操作

1） 详细内容

禁止对VIEW进行更新操作，如果需要更新，要对VIEW的数据源表进行直接更新

3-14 子查询相关

1） 详细内容

既可以用子查询（嵌套查询）又可以用结合查询的实现业务的时候，优先使用结合查询

例）

使用子查询

```
select EMP.WORKDEPT,EMP.EMPNO,EMP.FIRSTNME,EMP.SALARY
from EMPLOYEE as EMP
where EMP.WORKDEPT in (
        select DEP.DEPTNO
        from DEPARTMENT as DEP
        where DEP.ADMRDEPT = 'A00'
)
;
```

使用结合查询

```
select EMP.WORKDEPT,EMP.EMPNO,EMP.FIRSTNME,EMP.SALARY
from EMPLOYEE as EMP
inner join DEPARTMENT as DEP
   on EMP.WORKDEPT = DEP.DEPTNO
where DEP.ADMRDEPT = 'A00'
;
```

但是，当使用结合查询和使用子查询得到的结果集不一样的时候，本规则不可适用

（DEPARTMENT表中的DEPTNO不唯一的时候）

2） 理由

统一的SQL编写方式，会使SQL的解析更加高效

3） 特別事項

无特别事项

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌　CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新　CJB	新規作成者･会社略称	喬 恒斌　CJB	新規作成日	2016.04.18

3-14 子查询最多只能嵌套3层

1）　详细内容

· 子查询最多只能嵌套3层，并且，嵌套的时候，最底层的子查询的结果集要控制在2000件以下

· 在 SELECT的FROM文中编写嵌套查询的时候，要遵从如下形式

ORACLE :　(SELECT xxxx FROM TBLA WHERE A.COLE = 10) T1

MS SQL :　(SELECT xxxx FROM TBLA WHERE A.COLE = 10) AS T1

DB2　:　TABLE (SELECT xxxx FROM TBLA WHERE A.COLE = 10) T1

2）　理由

子查询的临时表没有索引，对临时表的再查询是全数据查询，因此，嵌套的子查询如果数据量过多，
很可能会影响其他的处理，必须要制定相关的限制

3）　特别事项

子查询的SQL编写要符合[2-2 保证SQL的可读性]的相关规约，子查询使用别名的时候，采用T#mn(m，n是从1开始的整数)
m，n以每各嵌套级别为单位进行切换，m为级，n为子查询序号（请参照下文示例）

例）

```
SELECT
   T1.EMPNO
   T1.DEPTNO
   T2.DEPTNAME
   T3.MAREANAME
FROM
   EMP T1,
   TABLE (SELECT
              T21.DEPTNO,
              T22.DEPTNAME
          FROM
              TENPO T21,
```

ﾀｲﾄﾙ:	社内共通規約−SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌　CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新　CJB	新規作成者･会社略称	喬 恒斌　CJB	新規作成日	2016.04.18

```
                DEPT T22
              WHERE
                T21.DEPTNO = T22.DEPTNO) T2
  ,TABLE (SELECT
                T31.DEPTNO,
                T32.MAREANAME
              FROM
                TENPO T31,
                DEPT T32,
              WHERE
                T31.DEPTNO = T32.DEPTNO) T3
WHERE
  T1.DEPTNO = T2.DERTNO
  AND T1.DEPTNO = T3.DEPTNO
```

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

4 针对索引（INDEX）的利用

4-1 有效利用复合列索引

1） 详细内容

在WHERE语句中或者JOIN语句的ON节指定已创建索引的列，可以在SQL执行的时候自动使用索引

（换句话说，可以为提高性能而建立WHERE语句中或者JOIN语句的ON节使用的列的索引）

· 在WHERE条件中，有索引的列的条件尽可能写在前面，无索引的列尽可能写到后面

· 如果没有适合的索引列，可以考虑追加索引，并且，唯一性越高的列，索引中越优先

· 追加索引的时候，优先对WHERE条件追加索引，然后再考虑ON条件的列是否需要追加索引

2） 理由

SQL与复合列索引的匹配度越高，查询就会越高效

例）

表T1(C1，C2，C3，C4)定义了复合列索引IX1(C1，C2，C3)

	条件	匹配度	说明	执行效率
1	WHERE C1='10' AND C2='20' AND C3='30'	3		最高的执行效率
2	WHERE C4='10'	0	符合索引中没有C4列	表扫描
3	WHERE C2='20'	0	C2之前的C1没有B	无法扫描到可用索引
4	WHERE C1='10'	1		索引扫描
5	WHERE C1>'20'	1		索引扫描
6	WHERE C1='10' AND C2='20'	2		索引扫描
7	WHERE C1='10' AND C2>'20'	2		索引扫描
8	WHERE C1='10' AND C3='30'	1	C3之前的C2没有	索引扫描
9	WHERE C1>'10' AND C2='20'	1	C2之前的C1是范围值	索引扫描

3） 特别事项

对于索引匹配度为0的SQL，因为会扫描全表和全索引，在性能上要格外注意。

4-2 WHERE条件的列与比较值的数据类型和长度

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

1） 详细内容

WHERE条件中的列同该列的比较值的数据类型和长度要一致

2） 理由

· 列和比较值之间的数据类型不同的时候，将无法使用该列的索引

例）

C2是SMALLINT数据类型，并且有索引

SELECT C1,C2 FROM TBL1 WHERE C2 > 11.2　无法使用索引

SELECT C1,C2 FROM TBL1 WHERE C2 > 17　　自动使用索引

· 列和比较值之间的长度不同的时候，又可能使用该列的索引

（类型是固定长字符串的场合，比较值长度小于等于列最大长度的时候是可以使用索引的）

3） 特别事项

无特别事项

4-3 WHERE条件中禁止进行算术演算

1） 详细内容

原则上即你知WHERE条件中禁止进行算术演算

2） 理由

· WEHRE条件中禁止进行算术演算的时候，将无法使用该列的索引

例）

C2是SMALLINT数据类型，并且有索引

SELECT C1,C2 FROM TBL1 WHERE C2 > 11.2　无法使用索引

SELECT C1,C2 FROM TBL1 WHERE C2 > 17　　自动使用索引

· 列和比较值之间的长度不同的时候，又可能使用该列的索引

（类型是固定长字符串的场合，比较值长度小于等于列最大长度的时候是可以使用索引的）

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

3） 特別事項

无特别事项

4-4 LIKE处理变更为BETWEEN IN处理

1） 详细内容

原则上不允许使用LIKE处理，若有LIKE处理应考虑时候可以变更为BETWEEN IN处理

例）

```
SELECT C1,C2 FROM TAB1
WHERE C2 LIKE 'BC%'          ← 第3个字符如果候选值可以确定是某个范围，
                               要变更为如下形式

SELECT C1,C2 FROM TAB1
WHERE C2 BETWEEN 'BC1' AND 'BC9'

SELECT C1,C2 FROM TAB1
WHERE C2 IN ('BC1','BC2','BC3','BC6','BC8')
```

2） 理由

能提高索引的命中率

3） 特別事項

a. 知道最大值和最小值的时候使用BETWEEN

b.值是枚举的时候使用IN

c.若使用LIKE有益于将来的系统维护的时候，可以使用LIKE

4-5 对同一个列进行复数个不等号条件并且存在OR关系的时候，要考虑改成UNION

1） 详细内容

对同一个列进行复数个不等号条件并且存在OR关系的时候，要考虑改成UNION

如果要同时使用ORDER BY，则无需遵从本规约

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌　CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新　CJB	新規作成者･会社略称	喬 恒斌　CJB	新規作成日	2016.04.18

2）　理由

存在对同一个列进行复数个不等号条件并且存在OR关系的时候，无法使用索引

（但是，但对同一个列进行复数各等号条件的时候，索引可以被调用）

例）

SELECT C1,C2 FROM TAB1 WHERE C3 = 80 OR C3 = 20

→ 索引(C3)会被调用

SELECT C1,C2 FROM TAB1 WHERE C3 > 80 OR C3 <= 20

→ 索引不会被使用，应进行如下变换

```
SELECT C1,C2 FROM TAB1 WHERE C3 > 80
UNION ALL
SELECT C1,C2 FROM TAB1 WHERE C3 <= 20
```

→ 索引(C3)会被调用

3）　特别事项

如果OR太多，这样的变换会增加UNION ALL的缩量，所以要根据实际情况进行斟酌

4-6

1）　详细内容

当IN的条件值过多的时候，要考虑该成BETWEEN

并且，IN的参数应控制在50各以内

2）　理由

IN的值过多的时候，有可能无法调用索引

3）　特别事项

无特别事项

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

4-7 结合条件指定的各表的列的数据类型和长度要相同

1） 详细内容

禁止使用数据类型和长度不同的项目作为ON条件进行结合处理

※当ON条件为复合条件的时候，有限考虑按此符合条件追加索引

2） 理由

ON条件的数据类型和长度不同的时候，无法调用索引

3） 特别事项

无特别事项

4-8 结合条件中禁止使用SUBSTR和CONCAT之类的函数

1） 详细内容

结合条件中禁止使用SUBSTR和CONCAT之类的函数

2） 理由

结合条件中使用SUBSTR和CONCAT等函数的时候，根据WHERE条件中指定的方法和函数的组合的不同，
有可能无法调用索引

3） 特别事项

无特别事项

ﾀｲﾄﾙ:	社内共通規約−SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

5 程序设计时的注意点

5-1 避免超时，死锁的基本考虑方式

1） 详细内容

对于ONLINE处理，为了尽可能在更新DB的时候减少竞争，只使用可实现业务的最小级别的锁

在处理中设定的锁，要尽可能的缩短锁时间

2） 理由

(1) 发生锁等待时候的影响

· ONLINE处理的时间和吞吐量会恶化

· 长时间的锁等待，会发生超时，导致系统发生异常

(2) 发生死锁时候的影响

· 发生死锁，导致系统发生异常

· 即使在发生超时/死锁的时候有自动ReTry的功能，系统的时间和吞吐量依然会恶化

3） 特别事项

(1) 防止超时的对策

· 尽可能减少锁的时间，当Batch和Online同时对一个表进行更新和读取，要调整BATCH的COMMIT时间

· 适当的对表进行行分割和列分割

(2) 防止死锁的对策

· 建立对数据表的访问顺序准则，读取和更新处理按照访问循序准则进行业务实现

5-2 对相当于主键的列进行变更的时候，不要使用UPDATE，要使用DELETE/INSERT

1） 详细内容

对主键或相当于主键的列进行变更的时候，不要使用UPDATE，要使用DELETE/INSERT

2） 理由

无特别事项

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

3） 特別事項

无特别事项

5-3 INSERT处理时候的主键重复判断，要交给DB进行判断

1） 详细内容

INSERT处理前不需要先SELECT出来进行重复判断，将重复判断处理交由INSERT处理

2） 理由

INSERT处理时的由DB自己判断已可以保证数据一致性，显式的SELECT处理是不必要的

3） 特別事項

无特别事项

5-4 对于字符列的排序

1） 详细内容

对于字符列的排序要按照UTF-8规则进行排序

2） 理由

无特别事项

3） 特別事項

无特别事项

5-5 INSERT处理禁止包含具有自增属性的列

1） 详细内容

INSERT处理禁止包含具有自增属性的列

2） 理由

自增属性的列有DB自身进行维护，以编程的方式改变会导致异常。、

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

3） 特別事項

需要注意当自增属性达到最大值以后的行为，是从新开始还是需要抛出异常。

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18

SQL规约检查清单

No	检查内容	参照章节
1	SQL文大小写文字与DB统一，关键字均为大写	2-2 保证SQL的可读性
2	SQL文换行正确，符合规约要求	2-2 保证SQL的可读性
3	SQL文中的空白，逗号的使用正确	2-2 保证SQL的可读性
4	否定形式的写法满足规约，没有使用某DBMS专有的语法	2-3 否定形式的编写方式
5	对CHAR型的日期项目的值有无判断没有使用NOT判断	2-4 日期项目的判定方式
6	SQL语句的长度在限定值之内	2-5 SQL的相关限制
7	SELECT 文没有使用SELECT *，对象字段都进行了明记	3-1 查询SQL中不可使用 SELECT *
8	没有使用SUM、,MAX、AVG、COUNT以外的集约函数	3-2 集约函数的有效利用
9	没有使用COUNT函数实现数据有无处理(DB2)	3-5 若只是为确认是否存在数据，不可使用COUNT（*）仅针对DB2
10	结合条件使用了索引列	3-8 如何进行高效的结合(JOIN)查询
11	JOIN控制在6个表以内	3-9 表结合数最大值
12	没有使用FULL OUTER JOIN	3-10 禁止使用FULL OUTER JOIN
13	WHERE条件中的列和比较值的类型，长度保持一致	4-2 WHERE条件的列与比较值的数据类型和长度
14	WHERE条件中没有算数运算处理	4-3 WHERE条件中禁止进行算术演算
15	LIKE文使用时，知道最大最小值的时候变更为BETWEEN，知道具体值的时候变更为IN	4-4 LIKE处理变更为BETWEEN IN处理
16	结合条件中没有使用SUBSTR和CONCAT之类的函数	4-8 结合条件中禁止使用SUBSTR和CONCAT之类的函数
17	JOIN结合时，表别名采用了T#(#是从1开始的整数)	3-15 子查询最多只能嵌套3层
18	子查询嵌套层没有超过3层	5-5 INSERT处理禁止包含具有自增属性的列
19	INSERT处理没有包含具有自增属性的列	
	<本清单根据项目团队的实际状况可以按需追加>	

ﾀｲﾄﾙ:	社内共通規約－SQL規約篇	Ver	最終承認日	2016.04.19	最終更新者･会社略称	喬 恒斌 CJB	最終更新日	2016.04.18
		1.00	最終承認者･会社略称	史 荣新 CJB	新規作成者･会社略称	喬 恒斌 CJB	新規作成日	2016.04.18
