第十五章 DNA 数据库

第一节 DNA 数据库的动机

最早我设计了数据库一直不想集成在养疗经[17]中进行发布，因为是明文存储，避免文字处理的结果被人肆意使用，因为文字结果没有技术性，也不是算法，花大量时间做这个还不能保密，令我头疼。于是有个最早的动机，用元基来加密。

现在我想把这个数据库的增删改查驱动都用肽展公式来计算实现，那就要完整的语义库，于是我有个强烈的动机就是完善它的基础。满足生化计算的逻辑底层知识来源保障。

第二节 DNA 数据库的应用需求

原文: 控制吸收

肽语: PEEOOUPOAVCUPOECAUPEDOAU

肽锁: MMAA

散列肽语 : MMAAPMMAAEMMAAEMMAAOMMAAOMMAAUMMAAPMMAAOMMAAAMMAAVMMAAC MMAAUMMAAPMMAAOMMAAEMMAACMMAAAMMAAUMMAAPMMAAEMMAADMMAAO MMAAAMMAAU

静态密钥: 0. 6/0. 3/0. 5/0. 632

A->MMVSVSPMMVSVSEMMVSVSEMMVSVSOMMVSVSOMMVSVSUMMVSVSPMMVSVSOMM
VSVSVSMMVSVSVMMVSVSCMMVSVSUMMVSVSPMMVSVSOMMVSVSEMMVSVSCMMVSVS
VSMMVSVSUMMVSVSPMMVSVSEMMVSVSDMMVSVSOMMVSVSVSMMVSVS
O->MMVSVSPMMVSVSEMMVSVSEMMVSVSESMMVSVSESMMVSVSUMMVSVSPMMVSVSES
MMVSVSVSMMVSVSVMMVSVSCMMVSVSUMMVSVSPMMVSVSESMMVSVSEMMVSVSCMMV
SVSVSMMVSVSUMMVSVSPMMVSVSEMMVSVSDMMVSVSESMMVSVSVSMMVSVS
P->MMVSVSECMMVSVSEMMVSVSEMMVSVSESMMVSVSESMMVSVSUMMVSVSECMMVSVSE
SMMVSVSVSMMVSVSVMMVSVSCMMVSVSUMMVSVSECMMVSVSESMMVSVSEMMVSVSCM

MVSVSVSMMVSVSUMMVSVSECMMVSVSEMMVSVSDMMVSVSESMMVSVSVSMMVSVS

M->CSCSVSVSECCSCSVSVSECSCSVSVSECSCSVSVSESCSCSVSVSESCSCSVSVSUCSCSVSVSE

CCSCSVSVSESCSCSVSVSVSCSCSVSVSVCSCSVSVSCCSCSVSVSUCSCSVSVSECCSCSVSVSESC

SCSVSVSECSCSVSVSCCSCSVSVSVSCSCSVSVSUCSCSVSVSECCSCSVSVSECSCSVSVSDCSCSV

SVSESCSCSVSVSVSCSCSVSVS

V->CSCSUQSUQSECCSCSUQSUQSECSCSUQSUQSECSCSUQSUQSESCSCSUQSUQSESCSCSUQS

UQSUCSCSUQSUQSECCSCSUQSUQSESCSCSUQSUQSUQSCSCSUQSUQSUQCSCSUQSUQSCCSC

SUQSUQSUCSCSUQSUQSECCSCSUQSUQSESCSCSUQSUQSECSCSUQSUQSCCSCSUQSUQSUQS

CSCSUQSUQSUCSCSUQSUQSECCSCSUQSUQSECSCSUQSUQSDCSCSUQSUQSESCSCSUQSUQS

UQSCSCSUQSUQS

E->CSCSUQSUQSUCCSCSUQSUQSDCSCSUQSUQSUCSCSUQSUQSDSCSCSUQSUQSUSCSCSUQS

UQSUCSCSUQSUQSDCCSCSUQSUQSUSCSCSUQSUQSUQSCSCSUQSUQSUQCSCSUQSUQSCCS

CSUQSUQSUCSCSUQSUQSUCCSCSUQSUQSUSCSCSUQSUQSUCSCSUQSUQSCCSCSUQSUQSU

QSCSCSUQSUQSUCSCSUQSUQSUCCSCSUQSUQSUCSCSUQSUQSDCSCSUQSUQSDSCSCSUQSU

QSUQSCSCSUQSUQS

C->IDSIDSUQSUQSUIDIDSIDSUQSUQSDIDSIDSUQSUQSUIDSIDSUQSUQSDSIDSIDSUQSUQSUS

IDSIDSUQSUQSUIDSIDSUQSUQSDIDIDSIDSUQSUQSUSIDSIDSUQSUQSUQSIDSIDSUQSUQSUQ

IDSIDSUQSUQSIDIDSIDSUQSUQSUIDSIDSUQSUQSUIDIDSIDSUQSUQSUSIDSIDSUQSUQSUIDS

IDSUQSUQSIDIDSIDSUQSUQSUQSIDSIDSUQSUQSUIDSIDSUQSUQSUIDIDSIDSUQSUQSUIDSID

SUQSUQSDIDSIDSUQSUQSDSIDSIDSUQSUQSUQSIDSIDSUQSUQS1000111110111111100101011000110011000111111101111111111010111100100111110111101001111101011111111011101111

S->IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQIU

QIDIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQIU

QIDQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQI

UIDIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQI

UIDQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQ

V->IDQIDIVIVIUIDIDQIDQVQVQDIDQIDIVQVQUIDQIDQVQVQDQIDIIDIVQVIVIDIIDQVQVIUI

DIIDIVQVQDIDIDIIDIVQVQUIIDIIDIVQVQVQIDQIDQVQVIVIDQIDQVQVQIDIDQIDQVQVQUID

QIDQVIVQUIDIDIIDQVQVQVIDIIDIVQVIUIDIIDQVQVQIDIDQIDQVIVQVQIDQIDQVIVQUIDIID
IVQVQUIDIDQIDQVQVIUIDQIDIVQVQDIDQIDQVQVQDQIDQIDIVQVQVQIDIIDQVQVQ
E->IEQIEIVIVIEIEIEQIEQVQVQEIEQIEIVQVQEIEQIEQVQVQEQIEIIEIVQVIVIEIIEQVQVIEIEIIEI
VQVQEIEIEIIEIVQVQEIIEIIEIVQVQVQIEQIEQVQVIVIEQIEQVQVQIEIEQIEQVQVQEIEQIEQVIV
QEIEIEIIEQVQVQVIEIIEIVQVIEIEIIEQVQVQIEIEQIEQVIVQVQIEQIEQVIVQEIEIIEIVQVQEIEIE
QIEQVQVIEIEQIEIVQVQEIEQIEQVQVQEQIEQIEIVQVQVQIEIIEQVQVQ
C->IEQIEIVIVIEIEIEQIEQVQVQEIEQIEIVQVQEIEQIEQVQVQEQIEIIEIVQVIVIEIIEQVQVIEIEIIEI
VQVQEIEIEIIEIVQVQEIIEIIEIVQVQVQIEQIEQVQVIVIEQIEQVQVQIEIEQIEQVQVQEIEQIEQVIV
QEIEIEIIEQVQVQVIEIIEIVQVIEIEIIEQVQVQIEIEQIEQVIVQVQIEQIEQVIVQEIEIIEIVQVQEIEIE
QIEQVQVIEIEQIEIVQVQEIEQIEQVQVQEQIEQIEIVQVQVQIEIIEQVQVQ
S->SESSESVSVSESESESSESVSVSESESSESVSVSESESSESVSVSESSESSESVSVSVSESSESVSVSES
ESSESVSVSESESESSESVSVSESSESSESVSVSVSSESSESVSVSVSESSESVSVSSESESSESVSVSESE
SSESVSVSESESESSESVSVSVSESSESVSVSESESSESVSVSSESESSESVSVSVSSESSESVSVSESESS
ESVSVSESESESSESVSVSESESSESVSVSESESSESVSVSESSESSESVSVSVSSESSESVSVS
A->SESSESAAESESESSESAAESESSESAAESESSESAAESSESSESAAAESSESAAESESSESAAESES
ESSESAAESSESSESAAASESSESAAAESSESAASESESSESAAESESSESAAESESESSESAAAESSES
AAESESSESAASESESSESAAASESSESAAESESSESAAESESESSESAAESESSESAAESESSESAAESS
ESSESAAASESSESAA
O->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
P->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
M->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAAS
OSOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAA
OOOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
静态肽展降元概率钥匙E: 1010101111110
静态肽展降元概率钥匙S:
1000111110111111100101011000110011000111111011111111110101111001001111101111010011111010
01111111101110111

静 态 肽 展 降 元 :

IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQIUQI
DIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQIUQI
DQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQIUI
DIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQIUI
DQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQU

静 态 肽 展 增 元 概 率 钥 匙 E:

00100000010000000100000010000000001001000001000000010010001000000000001

静 态 肽 展 增 元 概 率 钥 匙 S:

01000000101110100110101111000010000110000011000001100000111010110010111001011101010100001110000100001110010101101010100001100101100100110101111010011100011

静 态 肽 展 增 元 :

SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASOSO
AAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAOOO
SOAAOOSOAAOOSOAAOSOSOAAASOSOAAE

时间: 1613805008008

账号随机缓存字符串: ID0. 058573949085271804: 0. 11193692344279549

Session: SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASOSO
AAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAOOO
SOAAOOSOAAOOSOAAOSOSOAAASOSOAAE

==========

开始前序验证:

开 始 Session 解 析 :

SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASOSO
AAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAOOO
SOAAOOSOAAOOSOAAOSOSOAAASOSOAAE

开 始 概 率 钥 匙 解 析 :

10101011111101000111110111111100101011000110011000111111011111111110101111001001111101111010011111101011111111101110111001000000100000001000000010000000001001000001000000010010000

0100000000001010000001011101001101011110000100001100000110000011000001110101100101110010111101010100001110000100001110010101101010100001100101100100110101111010011100011

A->MMVSVSPMMVSVSEMMVSVSEMMVSVSOMMVSVSOMMVSVSUMMVSVSPMMVSVSOMMVSVSVSMMVSVSVMMVSVSCMMVSVSUMMVSVSPMMVSVSOMMVSVSEMMVSVSCMMVSVSVSMMVSVSUMMVSVSPMMVSVSEMMVSVSDMMVSVSOMMVSVSVSMMVSVS

O->MMVSVSPMMVSVSEMMVSVSEMMVSVSESMMVSVSESMMVSVSUMMVSVSPMMVSVSESMMVSVSVSMMVSVSVMMVSVSCMMVSVSUMMVSVSPMMVSVSESMMVSVSEMMVSVSCMMVSVSVSMMVSVSUMMVSVSPMMVSVSEMMVSVSDMMVSVSESMMVSVSVSMMVSVS

P->MMVSVSECMMVSVSEMMVSVSEMMVSVSESMMVSVSESMMVSVSUMMVSVSECMMVSVSESMMVSVSVSMMVSVSVMMVSVSCMMVSVSUMMVSVSECMMVSVSESMMVSVSEMMVSVSCMMVSVSVSMMVSVSUMMVSVSECMMVSVSEMMVSVSDMMVSVSESMMVSVSVSMMVSVS

M->CSCSVSVSECCSCSVSVSECSCSVSVSECSCSVSVSESCSCSVSVSESCSCSVSVSUCSCSVSVSECCSCSVSVSESCSCSVSVSVSCSCSVSVSVCSCSVSVSCCSCSVSVSUCSCSVSVSECCSCSVSVSECSCSVSVSCCSCSVSVSVSCSCSVSVSUCSCSVSVSECCSCSVSVSECSCSVSVSDCSCSVSVSESCSCSVSVSVSCSCSVSVS

V->CSCSUQSUQSECCSCSUQSUQSECSCSUQSUQSECSCSUQSUQSESCSCSUQSUQSESCSCSUQSUQSUCSCSUQSUQSECCSCSUQSUQSESCSCSUQSUQSUQSCSCSUQSUQSUQCSCSUQSUQSCCSCSUQSUQSUCSCSUQSUQSECCSCSUQSUQSESCSCSUQSUQSECSCSUQSUQSCCSCSUQSUQSUQSCSCSUQSUQSUCSCSUQSUQSECCSCSUQSUQSECSCSUQSUQSDCSCSUQSUQSESCSCSUQSUQSUQSCSCSUQSUQS

E->CSCSUQSUQSUCCSCSUQSUQSDCSCSUQSUQSUCSCSUQSUQSDSCSCSUQSUQSUSCSCSUQSUQSUCSCSUQSUQSUSCSCSUQSUQSDCCSCSUQSUQSUSCSCSUQSUQSUQSCSCSUQSUQSUQCSCSUQSUQSCCSCSUQSUQSUCSCSUQSUQSUCCSCSUQSUQSUSCSCSUQSUQSUCSCSUQSUQSCCSCSUQSUQSUQSCSCSUQSUQSUCSCSUQSUQSUCCSCSUQSUQSUCSCSUQSUQSDCSCSUQSUQSDSCSCSUQSUQSUQSCSCSUQSUQS

C->IDSIDSUQSUQSUIDIDSIDSUQSUQSDIDSIDSUQSUQSUIDSIDSUQSUQSDSIDSIDSUQSUQSUSIDSIDSUQSUQSUIDSIDSUQSUQSDIDIDSIDSUQSUQSUSIDSIDSUQSUQSUQSIDSIDSUQSUQSUQIDSIDSUQSUQSIDIDSIDSUQSUQSUIDSIDSUQSUQSUIDIDSIDSUQSUQSUSIDSIDSUQSUQSUIDSIDSUQSUQSUIDSIDSUQSUQSUIDSIDSUQSUQSUIDSID

SUQSUQSDIDSIDSUQSUQSDSIDSIDSUQSUQSUQSIDSIDSUQSUQS1000111110111111001010110
0011001100011111011111111110101111001001111101111010011111010111111101110111

S->IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQIU
QIDIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQIU
QIDQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQI
UIDIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQI
UIDQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQ

V->IDQIDIVIVIUIDIDQIDQVQVQDIDQIDIVQVQUIDQIDQVQVQDQIDIIDIVQVIVIDIIDQVQVIUI
DIIDIVQVQDIDIDIIDIVQVQUIIDIIDIVQVQVQIDQIDQVQVIVIDQIDQVQVQIDIDQIDQVQVQUID
QIDQVIVQUIDIDIIDQVQVQVIDIIDIVQVIUIDIIDQVQVQIDIDQIDQVIVQVQIDQIDQVIVQUIDIID
IVQVQUIDIDQIDQVQVIUIDQIDIVQVQDIDQIDQVQVQDQIDQIDIVQVQVQIDIIDQVQVQ
E->IEQIEIVIVIEIEIEQIEQVQVQEIEQIEIVQVQEIEQIEQVQVQEQIEIIEIVQVIVIEIIEQVQVIEIEIIEI
VQVQEIEIEIIEIVQVQEIIEIIEIVQVQVQIEQIEQVQVIVIEQIEQVQVQIEIEQIEQVQVQEIEQIEQVIV
QEIEIEIIEQVQVQVIEIIEIVQVIEIEIIEQVQVQIEIEQIEQVIVQVQIEQIEQVIVQEIEIIEIVQVQEIEIE
QIEQVQVIEIEQIEIVQVQEIEQIEQVQVQEQIEQIEIVQVQVQIEIIEQVQVQ
C->IEQIEIVIVIEIEIEQIEQVQVQEIEQIEIVQVQEIEQIEQVQVQEQIEIIEIVQVIVIEIIEQVQVIEIEIIEI
VQVQEIEIEIIEIVQVQEIIEIIEIVQVQVQIEQIEQVQVIVIEQIEQVQVQIEIEQIEQVQVQEIEQIEQVIV
QEIEIEIIEQVQVQVIEIIEIVQVIEIEIIEQVQVQIEIEQIEQVIVQVQIEQIEQVIVQEIEIIEIVQVQEIEIE
QIEQVQVIEIEQIEIVQVQEIEQIEQVQVQEQIEQIEIVQVQVQIEIIEQVQVQ
S->SESSESVSVSESESESSESVSVSESESSESVSVSESESSESVSVSESSESSESVSVSVSESSESVSVSES
ESSESVSVSESESESSESVSVSESSESSESVSVSVSSESSESVSVSVSESSESVSVSSESESSESVSVSESE
SSESVSVSESESESSESVSVSVSESSESVSVSESESSESVSVSSESESSESVSVSVSSESSESVSVSESESS
ESVSVSESESESSESVSVSESESSESVSVSESESSESVSVSESSESSESVSVSVSSESSESVSVS
A->SESSESAAESESESSESAAESESSESAAESESSESAAESSESSESAAAESSESAAESESSESAAESES
ESSESAAESSESSESAAASESSESAAAESSESAASESESSESAAESESSESAAESESESSESAAAESSES
AAESESSESAASESESSESAAASESSESAAESESSESAAESESESSESAAESESSESAAESESSESAAESS
ESSESAAASESSESAA
O->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO

SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO

OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA

P->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO

SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO

OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA

M->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAAS

OSOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAA

OOOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA

得到原降元元基DNA序列:

IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQIUQI

DIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQIUQI

DQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQIUI

DIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQIUI

DQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQU

得到新降元元基DNA序列:

IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQIUQI

DIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQIUQI

DQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQIUI

DIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQIUI

DQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQU

得到原元基DNA序列:

SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASOSO

AAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAOOO

SOAAOOSOAAOOSOAAOSOSOAAASOSOAAE

得到新元基DNA序列:

SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASOSO

AAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAOOO

SOAAOOSOAAOOSOAAOSOSOAAASOSOAAE

验证正确？

正确

==========

开始后序验证:

准 备 计 算 元 基 DNA 序 列 :

SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASOSO
AAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAOOO
SOAAOOSOAAOOSOAAOSOSOAAASOSOAAE

M->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAAS
OSOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAA
OOOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
P->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
O->SESSESAAESESESSESAAESESSESAAESESSESAAESSESSESAAAESSESAAESESSESAAESES
ESSESAAESSESSESAAASESSESAAAESSESAASESESSESAAESESSESAAESESESSESAAAESSES
AAESESSESAASESESSESAAASESSESAAESESSESAAESESESSESAAESESSESAAESESSESAAESS
ESSESAAASESSESAA
A->SESSESVSVSESESESSESVSVSESESSESVSVSESESSESVSVSESSESSESVSVSVSESSESVSVSE
SESSESVSVSESESESSESVSVSESSESSESVSVSVSSESSESVSVSVSESSESVSVSSESESSESVSVSES
ESSESVSVSESESESSESVSVSVSESSESVSVSESESSESVSVSSESESSESVSVSVSSESSESVSVSESES
SESVSVSESESESSESVSVSESESSESVSVSESESSESVSVSESSESSESVSVSVSSESSESVSVS0100000
0101110100110101111000010000110000011000001100000111010110010111001011101010100001110000
0100001110010101101010100001100101100100110101111010011100011 1

S->IEQIEIVIVIEIEIEQIEQVQVQEIEQIEIVQVQEIEQIEQVQVQEQIEIIEIVQVIVIEIIEQVQVIEIEIIEI
VQVQEIEIEIIEIVQVQEIIEIIEIVQVQVQIEQIEQVQVIVIEQIEQVQVQIEIEQIEQVQVQEIEQIEQVIV
QEIEIEIIEQVQVQVIEIIEIVQVIEIEIIEQVQVQIEIEQIEQVIVQVQIEQIEQVIVQEIEIIEIVQVQEIEIE
QIEQVQVIEIEQIEIVQVQEIEQIEQVQVQEQIEQIEIVQVQVQIEIIEQVQVQ

C->IEQIEIVIVIEIEIEQIEQVQVQEIEQIEIVQVQEIEQIEQVQVQEQIEIIEIVQVIVIEIIEQVQVIEIEIIEI

VQVQEIEIEIIEIVQVQEIIEIIEIVQVQVQIEQIEQVQVIVIEQIEQVQVQIEIEQIEQVQVQEIEQIEQVIV
QEIEIEIIEQVQVQVIEIIEIVQVIEIEIIEQVQVQIEIEQIEQVIVQVQIEQIEQVIVQEIEIIEIVQVQEIEIE
QIEQVQVIEIEQIEIVQVQEIEQIEQVQVQEQIEQIEIVQVQVQIEIIEQVQVQ
E->IDQIDIVIVIUIDIDQIDQVQVQDIDQIDIVQVQUIDQIDQVQVQDQIDIIDIVQVIVIDIIDQVQVIUI
DIIDIVQVQDIDIDIIDIVQVQUIIDIIDIVQVQVQIDQIDQVQVIVIDQIDQVQVQIDIDQIDQVQVQUID
QIDQVIVQUIDIDIIDQVQVQVIDIIDIVQVIUIDIIDQVQVQIDIDQIDQVIVQVQIDQIDQVIVQUIDIID
IVQVQUIDIDQIDQVQVIUIDQIDIVQVQDIDQIDQVQVQDQIDQIDIVQVQVQIDIIDQVQVQ
V->IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQI
UQIDIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQI
UQIDQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQ
IUIDIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQ
IUIDQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQ
得到原续降元元基DNA序列:
IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQIUQI
DIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQIUQI
DQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQIUI
DIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQIUI
DQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQU
得到后续降元元基DNA序列:
IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQIUQI
DIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQIUQI
DQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQIUI
DIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQIUI
DQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQU
开始整序验证:
准 备 计 算 元 基 DNA 序 列 :
SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASOSO

AAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAOOO
SOAAOOSOAAOOSOAAOSOSOAAASOSOAAE

V->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
E->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
C->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
S->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
A->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
O->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
P->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
M->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAAS
OSOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAA
OOOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
M->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAAS
OSOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAA
OOOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA

P->SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASO
SOAAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAO
OOSOAAOOSOAAOOSOAAOSOSOAAASOSOAA
O->SESSESAAESESESSESAAESESSESAAESESSESAAESSESSESAAAESSESAAESESSESAAESES
ESSESAAESSESSESAAASESSESAAAESSESAASESESSESAAESESSESAAESESESSESAAAESSES
AAESESSESAASESESSESAAASESSESAAESESSESAAESESESSESAAESESSESAAESESSESAAESS
ESSESAAASESSESAA
A->SESSESVSVSESESESSESVSVSESESSESVSVSESESSESVSVSESSESSESVSVSVSESSESVSVSE
SESSESVSVSESESESSESVSVSESSESSESVSVSVSSESSESVSVSVSESSESVSVSSESESSESVSVSES
ESSESVSVSESESESSESVSVSVSESSESVSVSESESSESVSVSSESESSESVSVSVSSESSESVSVSESES
SESVSVSESESESSESVSVSESESSESVSVSESESSESVSVSESSESSESVSVSVSSESSESVSVS0100000
0101110100110101111000010000110000011000001100000111010110010111001011101010100001110000
0100001110010101101010100001100101100100110101111010011100011 1

S->IEQIEIVIVIEIEIEQIEQVQVQEIEQIEIVQVQEIEQIEQVQVQEQIEIIEIVQVIVIEIIEQVQVIEIEIIEI
VQVQEIEIEIIEIVQVQEIIEIIEIVQVQVQIEQIEQVQVIVIEQIEQVQVQIEIEQIEQVQVQEIEQIEQVIV
QEIEIEIIEQVQVQVIEIIEIVQVIEIEIIEQVQVQIEIEQIEQVIVQVQIEQIEQVIVQEIEIIEIVQVQEIEIE
QIEQVQVIEIEQIEIVQVQEIEQIEQVQVQEQIEQIEIVQVQVQIEIIEQVQVQ

C->IEQIEIVIVIEIEIEQIEQVQVQEIEQIEIVQVQEIEQIEQVQVQEQIEIIEIVQVIVIEIIEQVQVIEIEIIEI
VQVQEIEIEIIEIVQVQEIIEIIEIVQVQVQIEQIEQVQVIVIEQIEQVQVQIEIEQIEQVQVQEIEQIEQVIV
QEIEIEIIEQVQVQVIEIIEIVQVIEIEIIEQVQVQIEIEQIEQVIVQVQIEQIEQVIVQEIEIIEIVQVQEIEIE
QIEQVQVIEIEQIEIVQVQEIEQIEQVQVQEQIEQIEIVQVQVQIEIIEQVQVQ
E->IDQIDIVIVIUIDIDQIDQVQVQDIDQIDIVQVQUIDQIDQVQVQDQIDIIDIVQVIVIDIIDQVQVIUI
DIIDIVQVQDIDIDIIDIVQVQUIIDIIDIVQVQVQIDQIDQVQVIVIDQIDQVQVQIDIDQIDQVQVQUID
QIDQVIVQUIDIDIIDQVQVQVIDIIDIVQVIUIDIIDQVQVQIDIDQIDQVIVQVQIDQIDQVIVQUIDIID
IVQVQUIDIDQIDQVQVIUIDQIDIVQVQDIDQIDQVQVQDQIDQIDIVQVQVQIDIIDQVQVQ
V->IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQI
UQIDIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQI
UQIDQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQ

IUIDIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQ

IUIDQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQ

得到原续降元元基DNA序列:

IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQIUQI

DIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQIUQI

DQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQIUI

DIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQIUI

DQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQU

得到后续降元元基DNA序列:

IDQIDIUQIUQIUIDIDQIDQUQQUQQDIDQIDIUQQUQQUIDQIDQUQQUQQDQIDIIDIUQQUQIUQI

DIIDQUQQUQIUIDIIDIUQQUQQDIDIDIIDIUQQUQQUIIDIIDIUQQUQQUQQIDQIDQUQQUQIUQI

DQIDQUQQUQQIDIDQIDQUQQUQQUIDQIDQUQIUQQUIDIDIIDQUQQUQQUQIDIIDIUQQUQIUI

DIIDQUQQUQQIDIDQIDQUQIUQQUQQIDQIDQUQIUQQUIDIIDIUQQUQQUIDIDQIDQUQQUQIUI

DQIDIUQQUQQDIDQIDQUQQUQQDQIDQIDIUQQUQQUQQIDIIDQUQQUQQU

验证正确?

正确

准 备 整 序 计 算 元 基 DNA 序 列 :

SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASOSO

AAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAOOO

SOAAOOSOAAOOSOAAOSOSOAAASOSOAAE

准 备 整 序 计 算 元 基 DNA 序 列 :

SOSOAAOOOSOAAOOSOAAOOSOAAOSOSOAAAOSOAAOOSOAAOOOSOAAOSOSOAAASOSO

AAAOSOAASOOSOAAOOSOAAOOOSOAAAOSOAAOOSOAASOOSOAAASOSOAAOOSOAAOOO

SOAAOOSOAAOOSOAAOSOSOAAASOSOAAE

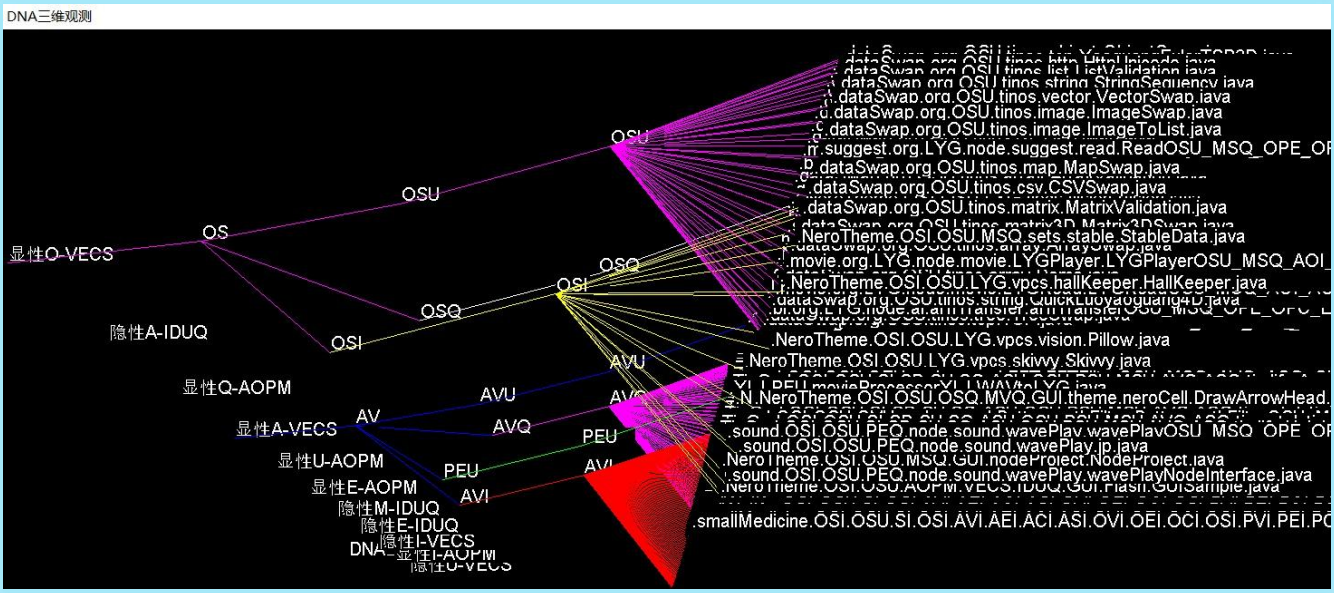
正确

第三节 DNA 数据库的具体描述

当数据原子能够完整的 无错的加密与解密, 同时, 密钥能完整固定不变, 密码在解密的过程中能完整还原则为实用. 于是我将概率钥匙和密钥对称性进行了离散推导如 **DNA 数据库数据加密**所示, 非常的庆幸我有自己的 Socket 流 PLSQL 数据库[4]可以用来做实验, 之后我的养疗经[17]将全面采用这种加密方式.

第四节 DNA 数据库的应用实现

DNA 数据库函数分类



DNA 数据库特征隐写

我在这里组织点文字来描述下什么是特征隐写, 这里可不是情报学的散列数据加密隐写在大文件里呀, 数据库特征隐写, 是将每一大段的文字进行元基翻译, 然后将元基按三元拆开来统计词汇, 这些词汇之后一来用与 DNN 分析, 二来做特征索引.

DNA 数据库文件安全 物理加密

之后养疗经[17]的所有文字文件都会变成物理元基加密文件. 用于本地存储, 提高一个安全级别, 因为, 明文文件存储信息非常不安全. 上面这图我做了个对比, 我把笔记原文和用法没有进行解密就显示出来, 大家可以看到明显的区别.

```
//元基编码加密:
String plsql= "setRoot: C: /DetaDB1; "; plsql+=
  "baseName: ZYY; ";
  plsql+= "tableName: zybc: insert; " +
          "columnValue: ID: "+ table. getValueAt(i, 0). toString(). replace(": ", "@biky@")+ "; " + "columnValue: 打  分  : "+ table.
          getValueAt(i, 1). toString(). replace(": ", "@biky@")+ "; " + "columnValue: 中药名称: "+ table. getValueAt(i, 2). toString().
          replace(": ", "@biky@")+ "; " + "columnValue: 笔记原文: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 3).
toString(). replace(": ", "@biky@"))+ "; " +
          "columnValue: 功效: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 4). toString(). replace(": ", "@biky@"))+ "; "
+
          "columnValue: 风险规避: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 5). toString(). replace(": ",
"@biky@"))+ "; " +
          "columnValue:  用量: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 6). toString(). replace(": ", "@biky@"))+ "; "
+
          "columnValue:  性味: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 7). toString(). replace(": ", "@biky@"))+ "; "
+
          "columnValue:  经脉: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 8). toString(). replace(": ", "@biky@"))+ "; "
+
          "columnValue: 中医馆药理: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 9). toString(). replace(": ",
"@biky@"))+ "; " +
          "columnValue:  经解: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 10). toString(). replace(": ", "@biky@"))+ ";
" +
          "columnValue:  崇源: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 11). toString(). replace(": ", "@biky@"))+ ";
" +
          "columnValue:  愚按: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 12). toString(). replace(": ", "@biky@"))+ ";
" +
          "columnValue:  搭配: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 13). toString(). replace(": ", "@biky@"))+ ";
" +
          "columnValue: 常见药: "+ new FullDNATokenPDI(). initonSect(table. getValueAt(i, 14).
```

toString(). replace(": ", "@biky@"))+ "; ";

try {

org. ME. plsql. db. plsql. imp. ExecPLSQLImp. *ExecPLSQL*(plsql, mod);

上面是一段我的数据库 SQL, 我目前没有将元基加密集成在数据库工程里, 现在这样调试已经成功. 下面是元基解密函数实例.

```
@SuppressWarnings({ "unchecked", "rawtypes" })
public Map<String, Object> listToMap(Map<String, Object> dic_yw, Map<String, Object> dic_li,
        Map<String, Object> dic_hai, Map<String, Object> dic_xz, Map<String, Object> dic_ya,
        Map<String, Object> dic_jm, Map<String, Object> dic_xw, Map<String, Object> dic_cy,
        Map<String, Object> dic_jj, Map<String, Object> dic_zf, Map<String, Object> dic_cj,
        Map<String, Object> dic_yl) {
    Map<String, Object> dic_map= new ConcurrentHashMap<String, Object>();
    Map<String, Object> map = null;
    //for(int i=0; i<)
    String plsql= "setRoot: C: /DetaDB1; " +
            "baseName: ZYY; " +
            "tableName: zybc: select; " +
            "condition: or: ID|<=|3000; ";
            //"condition: or: ID|==|2; ";
    try {
        map= org. ME. plsql. db. plsql. imp. ExecPLSQLImp. ExecPLSQL(plsql, true);
    }catch(Exception e1) { e1.
        printStackTrace();
    }
    ArrayList list= (ArrayList) map. get("obj");
    Iterator<HashMap<String, Object>> iterator= list. iterator();
    while(iterator. hasNext()) {
        HashMap<String, Object> hashmap= iterator. next();
        StringBuilder stringBuilder= new StringBuilder();
```

```
if(hashmap. containsKey("rowValue")) {
    HashMap<String, Object> rowValue= (HashMap<String, Object>) hashmap. get("rowValue");
    String keyName= null;
    if(rowValue. containsKey("中药名称")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("中药名称");
        keyName= temp. get("culumnValue"). toString();
        stringBuilder. append(temp. get("culumnValue"). toString());
    }
    if(rowValue. containsKey("常见药")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("常见药");
        String cj= null== temp. get("culumnValue")?"": temp. get("culumnValue"). toString();
        cj= new FullDNATokenPDI(). initonDeSect(cj);
        dic_cj. put(keyName, cj. replace("@biky@", ": "));
        stringBuilder. append(cj);
    }
    if(rowValue. containsKey("搭配")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("搭配");
        String zf= null== temp. get("culumnValue")?"": temp. get("culumnValue"). toString();
        zf= new FullDNATokenPDI(). initonDeSect(zf);
        dic_zf. put(keyName, zf. replace("@biky@", ": "));
        stringBuilder. append(zf);
    }
    if(rowValue. containsKey("愚按")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("愚按");
        String ya= null== temp. get("culumnValue")?"": temp. get("culumnValue"). toString();
        ya= new FullDNATokenPDI(). initonDeSect(ya);
        dic_ya. put(keyName, ya. replace("@biky@", ": "));
        stringBuilder. append(ya);
    }
    if(rowValue. containsKey("崇源")) {
```

```
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("崇源");
        String cy= null== temp. get("columnValue")?"": temp. get("columnValue"). toString();
        cy= new FullDNATokenPDI(). initonDeSect(cy);
        dic_cy. put(keyName, cy. replace("@biky@", ": "));
        stringBuilder. append(cy);
    }
    if(rowValue. containsKey("经解")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("经解");
        String jj= null== temp. get("columnValue")?"": temp. get("columnValue"). toString();
        jj= new FullDNATokenPDI(). initonDeSect(jj);
        dic_jj. put(keyName, jj. replace("@biky@", ": "));
        stringBuilder. append(jj);
    }
    if(rowValue. containsKey("经脉")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("经脉");
        String jm= null== temp. get("columnValue")?"": temp. get("columnValue"). toString();
        jm= new FullDNATokenPDI(). initonDeSect(jm);
        dic_jm. put(keyName, jm. replace("@biky@", ": "));
        stringBuilder. append(jm);
    }
    if(rowValue. containsKey("性味")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("性味");
        String xw= null== temp. get("columnValue")?"": temp. get("columnValue"). toString();
        xw= new FullDNATokenPDI(). initonDeSect(xw);
        dic_xw. put(keyName, xw. replace("@biky@", ": "));
        stringBuilder. append(xw);
    }
    if(rowValue. containsKey("用量")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("用量");
        String yl= null== temp. get("columnValue")?"": temp. get("columnValue"). toString();
```

```
 //yl= new FullDNATokenPDI(). initonDeSect(yl);
        dic_yl. put(keyName, yl. replace("@biky@", ": "));
        stringBuilder. append(yl);
    }
    if(rowValue. containsKey("风险规避")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("风险规避");
        String hai= null== temp. get("columnValue")?"": temp. get("columnValue"). toString();
     hai= new FullDNATokenPDI(). initonDeSect(hai);
        dic_hai. put(keyName, hai. replace("@biky@", ": "));
        stringBuilder. append(hai);
    }
    if(rowValue. containsKey("功效")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("功效");
        String li= null== temp. get("columnValue")?"": temp. get("columnValue"). toString();
     li= new FullDNATokenPDI(). initonDeSect(li); dic_li.
        put(keyName, li. replace("@biky@", ": "));
        stringBuilder. append(li);
    }
    if(rowValue. containsKey("笔记原文")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("笔记原文");
        String yw= null== temp. get("columnValue")?"": temp. get("columnValue"). toString();
        //yw= new FullDNATokenPDI(). initonDeSect(yw);
        dic_yw. put(keyName, yw. replace("@biky@", ": "));
        stringBuilder. append(yw);
    }
    if(rowValue. containsKey("中医馆药理")) {
        HashMap<String, Object> temp= (HashMap<String, Object>) rowValue. get("中医馆药理");
        String xz= null== temp. get("columnValue")?"": temp. get("columnValue"). toString();
     xz= new FullDNATokenPDI(). initonDeSect(xz);
        dic_xz. put(keyName, xz. replace("@biky@", ": "));
```

```
                    stringBuilder. append(xz);
                }
                dic_map. put(keyName, stringBuilder. toString(). replace("@biky@", ": "));
            }
        }
        return dic_map;
    }
```

上面源码的作用是将文字进行元基化编辑, 编辑的过程比较机械, 仅仅通过 ASCII 编码进行 char 数字化然后元基编码的逻辑, 之后随着语料库的不断完善, 我会逐渐将其描述成语义元基编码方式, 方便肽展计算.

DNA 数据库数据加密

	AOPM	VECS	IDUQ
A -> B	TRUE	TRUE	TRUE
B -> C	TRUE	TRUE	TRUE
A != C	FALSE	FALSE	FASLE
C -> B	TRUE	TRUE	TRUE

今天来组织点文字描述下这个逻辑: A 为待加密文件, B 为肽展公式计算的降元级别, C 为肽展公式计算的增元级别. 通过计算会发现一次降元和一次增元计算后, 文件与结果已经不对应.

XOR

	A	B	C
A -> B	AOPM VECS		
B -> C		VECSAOPM	
A != C			
C -> B			MPOA SCEV

于是我开始进行离散逻辑推导归纳, 发现MPOASCEV 的肽展计算能让C 与 B 计算结果吻合. 但 A 还是不等于 C

XNOR

	A	B	C
A -> B	MPOA SCEV		
B -> C		VECSAOPM	
A != C			
C -> B			MPOA SCEV

而反向的 A 进行 MPOASCEV 肽展计算, A 也是不等于 C, 于是我开始思考是函数的问题还是概率的问题. 于是进行概率钥匙推导计算 Token 元基概率钥匙推导如下:

	A		B		C	
A -> B	AOPMVECS	FD				
B -> C			VECSAOPM	FI		
A -> B	AOPMVECS	TD				
B -> C			VECSAOPM	TI		
C -> B					MPOASCEV	TI
B->C			VECSAOPM	TI		
C=A -> B	MPOASCEV	TI				

通过测试研究发现, 我得到一个结论, A 到 B 的时候, 已经将元基中的组合拓扑不稳定的元基过滤了, 于是B 到 C 和 C 到 B 的过程是一个循环可破解的过程, 于是我继续跟进测试, 得到健全的列表归纳如下:

Token 元基概率钥匙归纳如下:

	DNA		PDE B		PDE C	
Dna	AOPMVECS	TDD	VECSAOPM	TII		
Full dna					MPOASCEV	TDI
Full back dna			MPOASCEV	TDI	VECSAOPM	TII
Full up dna			VECSAOPM	TII	MPOASCEV	TDI

在上面表中, 2 组概率钥匙 D, I, D 用于服务器端计算, I 用于客户端通讯.

目前已经在养疗经[17]中实用, 效果不错. **T 为带钥匙计算， 中间的的 DI 为增元降元计算识别，右边 DI 为钥匙种类。**

早期的肽展公式我没有单独量化, 可以见 git 的 catalytic 最早肽展计算, 24 个分类能依次展示, 现在我把其概率串量化的版本这里也展示出来, 保证演化的完整性.

```
package OSI. OSU. SI. SD. SU. SQ. ASU. OSU. PSU. MSU. AVQ. ASQ. tin. catalytic. procedure. pde;
//注意: 该 文件对应的是罗瑶光先生 DNA 编码 与 计算的两本 国家软著作 思想的编码 实现.
//公安部 与 知识产权委员会 已经备案, 可阅读 相关 著作权 原文 进行逻辑辨别.
public class PDE_Formular {
    public static TokenPDI tokenPDI;
    public static void main(String[] argv) { Initon
        initonA= new Initon(); initonA. setM(); // 改
        成 O 测试下Initon initonV= new Initon();
        initonV. setS();
        Initon initonS= new Initon(); initonS.
        setI();
```

```
Initon initonS1= new Initon(); initonS1. setO();
Initon initonS2= new Initon(); initonS2. setC();
Initon initonS3= new Initon(); initonS3. setU();
Initon initonS4= new Initon(); initonS4. setO();
Initon initonS5= new Initon(); initonS5. setC();
Initon initonS6= new Initon();
initonS6. setI();
Initon initonS7= new Initon(); initonS7. setP();
Initon initonS8= new Initon(); initonS8. setC();
Initon initonS9= new Initon(); initonS9. setU();
Initon initonS10= new Initon(); initonS10.
setP();
Initon initonS11= new Initon(); initonS11.
setC();
Initon initonS12= new Initon();
initonS12. setI();
initonA.      next=      initonV;
initonV.      prev=      initonA;
initonV.      next=      initonS;
initonS.      prev=      initonV;
initonS.      next=      initonS1;
initonS1. prev= initonS;
```

```
        initonS. next= initonS1; initonS1.
        prev= initonS; initonS1. next=
        initonS2; initonS2. prev= initonS1;
        initonS2. next= initonS3; initonS3.
        prev= initonS2; initonS3. next=
        initonS4; initonS4. prev= initonS3;
        initonS4. next= initonS5; initonS5.
        prev= initonS4; initonS5. next=
        initonS6; initonS6. prev= initonS5;
        initonS6. next= initonS7; initonS7.
        prev= initonS6; initonS7. next=
        initonS8; initonS8. prev= initonS7;
        initonS8. next= initonS9; initonS9.
        prev= initonS8; initonS9. next=
        initonS10; initonS10. prev=
        initonS9; initonS10. next=
        initonS11; initonS11. prev=
        initonS10; initonS11. next=
        initonS12; initonS12. prev=
        initonS11;
        PDE_Formular pDE_RNA_Formular = new PDE_Formular();
        pDE_RNA_Formular. do_PDE_RNA_Formular(initonA);
    }
    public void do_PDE_RNA_Formular(Initon initon) {
        tokenPDI= new TokenPDI();
        tokenPDI. key= new double[4];
```

```
tokenPDI. key[0]= 0. 6;
tokenPDI. key[1]= 0. 3;
tokenPDI. key[2]= 0. 5;
tokenPDI. key[3]= 0. 632;
//初始
Initon InitonPDE= initon;
System. out. print("input: " + InitonPDE. getStore());
while(InitonPDE. hasNext()) {
    InitonPDE= InitonPDE. forwardNext(); System. out.
    print(InitonPDE. getStore());
}
System. out. println();
System. out. println("肽展  降元");
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
InitonLinkDNA initonLinkDNA= new InitonLinkDNA();
initonLinkDNA. setInitonLink(InitonPDE);
InitonPDE= new PDE_Decrement_Formular(). PDE_DecrementA(initonLinkDNA);
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
System. out. println();
System. out. println("降元A = V + S");
while(InitonPDE. hasNext()) {
    System. out. print(InitonPDE. getStore());
    InitonPDE= InitonPDE. forwardNext();
}
System. out. print(InitonPDE. getStore());
while(InitonPDE. hasPrev()) {
```

```
        InitonPDE= InitonPDE. forwardPrev();
    }
    initonLinkDNA. setInitonLink(InitonPDE);
    InitonPDE= new PDE_Decrement_Formular(). PDE_DecrementO(initonLinkDNA);
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println("降元O = E + S");
    while(InitonPDE. hasNext()) {
        System. out. print(InitonPDE. getStore());
        InitonPDE= InitonPDE. forwardNext();
    }
    System. out. print(InitonPDE. getStore());
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    initonLinkDNA. setInitonLink(InitonPDE);
    InitonPDE= new PDE_Decrement_Formular(). PDE_DecrementP(initonLinkDNA);
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println("降元P = E + C");
    while(InitonPDE. hasNext()) {
        System. out. print(InitonPDE. getStore());
        InitonPDE= InitonPDE. forwardNext();
    }
    System. out. print(InitonPDE. getStore());
    while(InitonPDE. hasPrev()) {
```

```
        InitonPDE= InitonPDE. forwardPrev();
    }
    initonLinkDNA. setInitonLink(InitonPDE);
    InitonPDE= new PDE_Decrement_Formular(). PDE_DecrementM(initonLinkDNA);
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println("降元M = C + S");
    while(InitonPDE. hasNext()) {
        System. out. print(InitonPDE. getStore());
        InitonPDE= InitonPDE. forwardNext();
    }
    System. out. print(InitonPDE. getStore());
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    initonLinkDNA. setInitonLink(InitonPDE);
    InitonPDE= new PDE_Decrement_Formular(). PDE_DecrementV(initonLinkDNA);
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println("降元V = U + Q");
    while(InitonPDE. hasNext()) {
        System. out. print(InitonPDE. getStore());
        InitonPDE= InitonPDE. forwardNext();
    }
    System. out. print(InitonPDE. getStore());
    while(InitonPDE. hasPrev()) {
```

```
        InitonPDE= InitonPDE. forwardPrev();
    }
    initonLinkDNA. setInitonLink(InitonPDE);
    Initon InitonPDE_COPY= InitonPDE. copyRNA(InitonPDE); System. out.
    println();
    System. out. println("1降元概率IU");
    doE_IU(InitonPDE, initonLinkDNA);

    initonLinkDNA. setInitonLink(InitonPDE_COPY); System. out.
    println();
    System. out. println("2降元概率DU");
    doE_DU(InitonPDE_COPY, initonLinkDNA);
}
private static void doE_IU(Initon InitonPDE, InitonLinkDNA initonLinkDNA) {
    InitonPDE= new PDE_Decrement_Formular(). PDE_DecrementE_IU(initonLinkDNA, tokenPDI,
false);
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println("11降元E = I + U");
    while(InitonPDE. hasNext()) {
        System. out. print(InitonPDE. getStore());
        InitonPDE= InitonPDE. forwardNext();
    }
    System. out. print(InitonPDE. getStore());
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    initonLinkDNA. setInitonLink(InitonPDE);
```

```
InitonPDE= new PDE_Decrement_Formular(). PDE_DecrementC(initonLinkDNA);
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
System. out. println();
System. out. println("11降元C = I + D");
while(InitonPDE. hasNext()) {
    System. out. print(InitonPDE. getStore());
    InitonPDE= InitonPDE. forwardNext();
}
System. out. print(InitonPDE. getStore());
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
initonLinkDNA. setInitonLink(InitonPDE);
initonLinkDNA. setInitonLink(InitonPDE);
Initon InitonPDE_COPYSI= InitonPDE. copyRNA(InitonPDE); Initon
InitonPDE_COPYSQ= InitonPDE. copyRNA(InitonPDE); System. out. println();
String s= "1111概率S_";
String si= "1111概率S_I_"; String
sq= "1111概率S_Q_";
System. out. println("1111概率S");
doS(s, InitonPDE, initonLinkDNA);
initonLinkDNA. setInitonLink(InitonPDE_COPYSI); System. out.
println();
System. out. println("1112概率S_I");
doS_I(si, InitonPDE_COPYSI, initonLinkDNA); initonLinkDNA.
setInitonLink(InitonPDE_COPYSQ); System. out. println();
```

```
        System. out. println("1113概率S_Q");
        doS_Q(sq, InitonPDE_COPYSQ, initonLinkDNA);
    }

    private static void doS_Q(String sq, Initon InitonPDE, InitonLinkDNA initonLinkDNA) { InitonPDE= new
        PDE_Decrement_Formular(). PDE_DecrementS_Q(initonLinkDNA, tokenPDI); while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        System. out. println();


        System. out. println(sq+ "降元S = Q");
        while(InitonPDE. hasNext()) {
            System. out. print(InitonPDE. getStore());
            InitonPDE= InitonPDE. forwardNext();
        }
        System. out. print(InitonPDE. getStore());
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);

        //全部 收
        System. out. println();
        System. out. println(sq+ "肽展  增元");
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA= new InitonLinkDNA();
        initonLinkDNA. setInitonLink(InitonPDE);
```

```
        InitonPDE= new PDE_Increment_Formular(). PDE_IncrementV(initonLinkDNA);
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        System. out. println();
        System. out. println(sq+ "V = U + Q");
        while(InitonPDE. hasNext()) {
            System. out. print(InitonPDE. getStore());
            InitonPDE= InitonPDE. forwardNext();
        }
        System. out. print(InitonPDE. getStore());
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);
        Initon InitonPDE_COPY= InitonPDE. copyRNA(InitonPDE); System. out.
        println();
        System. out. println(sq+ "概率Increment IU");
        doIncrementE_IU(sq, InitonPDE, initonLinkDNA);

        initonLinkDNA. setInitonLink(InitonPDE_COPY); System. out.
        println();
        System. out. println(sq+ "概率Increment DU");
        doIncrementE_DU(sq, InitonPDE_COPY, initonLinkDNA);
    }

    private static void doS_I(String si, Initon InitonPDE, InitonLinkDNA initonLinkDNA) { InitonPDE= new
        PDE_Decrement_Formular(). PDE_DecrementS_I(initonLinkDNA, tokenPDI,
false);
        while(InitonPDE. hasPrev()) {
```

```
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println(si+ "11降元S = I");
    while(InitonPDE. hasNext()) {
        System. out. print(InitonPDE. getStore());
        InitonPDE= InitonPDE. forwardNext();
    }
    System. out. print(InitonPDE. getStore());
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    initonLinkDNA. setInitonLink(InitonPDE);

    //全部 收
    System. out. println();
    System. out. println(si+ "11肽展 增元");
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    initonLinkDNA= new InitonLinkDNA();
    initonLinkDNA. setInitonLink(InitonPDE);
    InitonPDE= new PDE_Increment_Formular(). PDE_IncrementV(initonLinkDNA);
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println(si+ "11V = U + Q");
    while(InitonPDE. hasNext()) {
        System. out. print(InitonPDE. getStore());
```

```
            InitonPDE= InitonPDE. forwardNext();
        }
        System. out. print(InitonPDE. getStore());
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);
        Initon InitonPDE_COPY= InitonPDE. copyRNA(InitonPDE); System. out.
        println();
        System. out. println(si+ "111概率Increment IU");
        doIncrementE_IU(si, InitonPDE, initonLinkDNA);

        initonLinkDNA. setInitonLink(InitonPDE_COPY); System. out.
        println();
        System. out. println(si+ "112概率Increment DU");
        doIncrementE_DU(si, InitonPDE_COPY, initonLinkDNA);
    }

    private static void doS(String s, Initon InitonPDE, InitonLinkDNA initonLinkDNA) { InitonPDE= new
        PDE_Decrement_Formular(). PDE_DecrementS(initonLinkDNA); while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        System. out. println();

        System. out. println(s+ "11降元S = I + Q");
        while(InitonPDE. hasNext()) {
            System. out. print(InitonPDE. getStore());
            InitonPDE= InitonPDE. forwardNext();
        }
```

```
System. out. print(InitonPDE. getStore());
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
initonLinkDNA. setInitonLink(InitonPDE);

//全部 收
System. out. println();
System. out. println(s+ "11肽展 增元");
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
initonLinkDNA= new InitonLinkDNA();
initonLinkDNA. setInitonLink(InitonPDE);
InitonPDE= new PDE_Increment_Formular(). PDE_IncrementV(initonLinkDNA);
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
System. out. println();
System. out. println(s+ "11V = U + Q");
while(InitonPDE. hasNext()) {
    System. out. print(InitonPDE. getStore());
    InitonPDE= InitonPDE. forwardNext();
}
System. out. print(InitonPDE. getStore());
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
initonLinkDNA. setInitonLink(InitonPDE);
Initon InitonPDE_COPY= InitonPDE. copyRNA(InitonPDE);
```

```
        System. out. println();
        System. out. println(s+ "111概率Increment IU");
        doIncrementE_IU(s, InitonPDE, initonLinkDNA);

        initonLinkDNA. setInitonLink(InitonPDE_COPY); System. out.
        println();
        System. out. println(s+ "112概率Increment DU");
        doIncrementE_DU(s, InitonPDE_COPY, initonLinkDNA);
    }

    private static void doE_DU(Initon InitonPDE, InitonLinkDNA initonLinkDNA) { InitonPDE= new
        PDE_Decrement_Formular(). PDE_DecrementE_DU(initonLinkDNA,
tokenPDI);
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        System. out. println();
        System. out. println("21降元E = D + U");
        while(InitonPDE. hasNext()) {
            System. out. print(InitonPDE. getStore());
            InitonPDE= InitonPDE. forwardNext();
        }
        System. out. print(InitonPDE. getStore());
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);
        InitonPDE= new PDE_Decrement_Formular(). PDE_DecrementC(initonLinkDNA);
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
```

```
}
System. out. println();
System. out. println("21降元C = I + D");
while(InitonPDE. hasNext()) {
    System. out. print(InitonPDE. getStore());
    InitonPDE= InitonPDE. forwardNext();
}
System. out. print(InitonPDE. getStore());
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}

initonLinkDNA. setInitonLink(InitonPDE);
Initon InitonPDE_COPYSI= InitonPDE. copyRNA(InitonPDE); Initon
InitonPDE_COPYSQ= InitonPDE. copyRNA(InitonPDE); String s= "2222概率
S_";
String si= "2222概率S_I_";
String sq= "2222概率S_Q_"; System.
out. println();
System. out. println("2111概率S");
doS(s, InitonPDE, initonLinkDNA);

initonLinkDNA. setInitonLink(InitonPDE_COPYSI); System. out.
println();
System. out. println("2112概率S_I");
doS_I(si, InitonPDE_COPYSI, initonLinkDNA);

initonLinkDNA. setInitonLink(InitonPDE_COPYSQ); System. out.
println();
System. out. println("2113概率S_Q");
```

```
        doS_Q(sq, InitonPDE_COPYSQ, initonLinkDNA);
    }

    private static void doIncrementE_DU(String sq, Initon InitonPDE, InitonLinkDNA initonLinkDNA) { initonLinkDNA.
        setInitonLink(InitonPDE);
        InitonPDE= new PDE_Increment_Formular(). PDE_IncrementE_DU(initonLinkDNA);
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        System. out. println();
        System. out. println(sq+ "E = D + U");
        while(InitonPDE. hasNext()) {
            System. out. print(InitonPDE. getStore());
            InitonPDE= InitonPDE. forwardNext();
        }
        System. out. print(InitonPDE. getStore());
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);
        InitonPDE= new PDE_Increment_Formular(). PDE_IncrementC(initonLinkDNA);
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        System. out. println();
        System. out. println(sq+ "C = I + D");
        while(InitonPDE. hasNext()) {
            System. out. print(InitonPDE. getStore());
            InitonPDE= InitonPDE. forwardNext();
        }
```

```
        System. out. print(InitonPDE. getStore());
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);

        initonLinkDNA. setInitonLink(InitonPDE);
        Initon InitonPDE_COPYSI= InitonPDE. copyRNA(InitonPDE); Initon
        InitonPDE_COPYSQ= InitonPDE. copyRNA(InitonPDE);

        doIncrementS(sq + "EDU_IQ_", InitonPDE, initonLinkDNA);
        doIncrementS_I(sq + "EDU_I_", InitonPDE_COPYSI, initonLinkDNA);
        doIncrementS_Q(sq + "EDU_Q_", InitonPDE_COPYSQ, initonLinkDNA);
    }

    private static void doIncrementE_IU(String s, Initon InitonPDE, InitonLinkDNA initonLinkDNA) { initonLinkDNA.
        setInitonLink(InitonPDE);
        InitonPDE= new PDE_Increment_Formular(). PDE_IncrementE_IU(initonLinkDNA, tokenPDI,
false);
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        System. out. println();
        System. out. println(s+ "E = I + U");
        while(InitonPDE. hasNext()) {
            System. out. print(InitonPDE. getStore());
            InitonPDE= InitonPDE. forwardNext();
        }
        System. out. print(InitonPDE. getStore());
        while(InitonPDE. hasPrev()) {
```

```
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);
        InitonPDE= new PDE_Increment_Formular(). PDE_IncrementC(initonLinkDNA);
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        System. out. println();
        System. out. println(s+ "C = I + D");
        while(InitonPDE. hasNext()) {
            System. out. print(InitonPDE. getStore());
            InitonPDE= InitonPDE. forwardNext();
        }
        System. out. print(InitonPDE. getStore());
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);
        InitonPDE= new PDE_Increment_Formular(). PDE_IncrementS(initonLinkDNA);
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);
        Initon InitonPDE_COPYSI= InitonPDE. copyRNA(InitonPDE); Initon
        InitonPDE_COPYSQ= InitonPDE. copyRNA(InitonPDE);

        doIncrementS(s + "EIU_IQ_", InitonPDE, initonLinkDNA);
        doIncrementS_I(s + "EIU_I_", InitonPDE_COPYSI, initonLinkDNA);
        doIncrementS_Q(s + "EIU_Q_", InitonPDE_COPYSQ, initonLinkDNA);
    }
```

```
private static void doIncrementS_Q(String s, Initon InitonPDE, InitonLinkDNA initonLinkDNA) { initonLinkDNA.
    setInitonLink(InitonPDE);
    InitonPDE= new PDE_Increment_Formular(). PDE_IncrementS_Q(initonLinkDNA);
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println(s+ "S = Q");
    while(InitonPDE. hasNext()) {
        System. out. print(InitonPDE. getStore());
        InitonPDE= InitonPDE. forwardNext();
    }
    System. out. print(InitonPDE. getStore());
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    initonLinkDNA. setInitonLink(InitonPDE);
    doIncrementAOPM(s, InitonPDE, initonLinkDNA);
}

private static void doIncrementS_I(String s, Initon InitonPDE, InitonLinkDNA initonLinkDNA) { initonLinkDNA.
    setInitonLink(InitonPDE);
    InitonPDE= new PDE_Increment_Formular(). PDE_IncrementS_I(initonLinkDNA, tokenPDI);
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println(s+ "S = I");
    while(InitonPDE. hasNext()) {
```

```
            System. out. print(InitonPDE. getStore());
            InitonPDE= InitonPDE. forwardNext();
        }
        System. out. print(InitonPDE. getStore());
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);
        doIncrementAOPM(s, InitonPDE, initonLinkDNA);
    }

    private static void doIncrementS(String s, Initon InitonPDE, InitonLinkDNA initonLinkDNA) { initonLinkDNA.
        setInitonLink(InitonPDE);
        InitonPDE= new PDE_Increment_Formular(). PDE_IncrementS(initonLinkDNA);
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        System. out. println();
        System. out. println(s+ "S = I + Q");
        while(InitonPDE. hasNext()) {
            System. out. print(InitonPDE. getStore());
            InitonPDE= InitonPDE. forwardNext();
        }
        System. out. print(InitonPDE. getStore());
        while(InitonPDE. hasPrev()) {
            InitonPDE= InitonPDE. forwardPrev();
        }
        initonLinkDNA. setInitonLink(InitonPDE);
        doIncrementAOPM(s, InitonPDE, initonLinkDNA);
    }
```

```
private static void doIncrementAOPM(String s, Initon InitonPDE, InitonLinkDNA initonLinkDNA) { InitonPDE= new
    PDE_Increment_Formular(). PDE_IncrementA(initonLinkDNA); while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println(s+ "A = V + S");
    while(InitonPDE. hasNext()) {
        System. out. print(InitonPDE. getStore());
        InitonPDE= InitonPDE. forwardNext();
    }
    System. out. print(InitonPDE. getStore());
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    initonLinkDNA. setInitonLink(InitonPDE);
    InitonPDE= new PDE_Increment_Formular(). PDE_IncrementO(initonLinkDNA);
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
    }
    System. out. println();
    System. out. println(s+ "O = E + S");
    while(InitonPDE. hasNext()) {
        System. out. print(InitonPDE. getStore());
        InitonPDE= InitonPDE. forwardNext();
    }
    System. out. print(InitonPDE. getStore());
    while(InitonPDE. hasPrev()) {
        InitonPDE= InitonPDE. forwardPrev();
```

```
}
initonLinkDNA. setInitonLink(InitonPDE);
InitonPDE= new PDE_Increment_Formular(). PDE_IncrementP(initonLinkDNA);
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
System. out. println();
System. out. println(s+ "P = E + C");
while(InitonPDE. hasNext()) {
    System. out. print(InitonPDE. getStore());
    InitonPDE= InitonPDE. forwardNext();
}
System. out. print(InitonPDE. getStore());
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
initonLinkDNA. setInitonLink(InitonPDE);
InitonPDE= new PDE_Increment_Formular(). PDE_IncrementM(initonLinkDNA);
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
}
System. out. println();
System. out. println(s+ "M = C + S");
while(InitonPDE. hasNext()) {
    System. out. print(InitonPDE. getStore());
    InitonPDE= InitonPDE. forwardNext();
}
System. out. print(InitonPDE. getStore());
while(InitonPDE. hasPrev()) {
    InitonPDE= InitonPDE. forwardPrev();
```

```
            }
            initonLinkDNA. setInitonLink(InitonPDE);
        }
}
```

```
//上面文件测试结果如下:
  input: MSIOCUOCIPCUPCI
肽展 降元
降 元 A = V + S
MSIOCUOCIPCUPCI 降 元
O = E + S
MSIESCUESCIPCUPCI 降元
P = E + C
MSIESCUESCIECCUECCI
降 元 M = C + S
CSSIESCUESCIECCUECCI 降 元
V = U + Q
CSSIESCUESCIECCUECCI
1降元概率IU

11 降 元 E = I + U
CSSIIUSCUDUSCIDUCCUIUCCI 11降元
C = I + D
IDSSIIUSIDUDUSIDIDUIDIDUIUIDIDI
1111概率S

1111 概 率 S_11 降 元 S = I + Q
IDIQIQIIUIQIDUDUIQIDIDUIDIDUIUIDIDI
1111概率S_11肽展 增元
```

1111 概 率 S_11V = U + Q

IDIQIQIIUIQIDUDUIQIDIDUIDIDUIUIDIDI

1111概率S_111概率Increment IU

1111 概 率 S_E = I + U

IDIQIQIEIQIEEIQIDIEIDIEEIDIDI 1111 概

率 S_C = I + D

CIQIQIEIQIEEIQCIECIEECCI 1111 概 率

S_EIU_IQ_S = I + Q

CSSIESIEESCIECIEECCI

1111概率S_EIU_IQ_A = V + S

CSSIESIEESCIECIEECCI 1111概率

S_EIU_IQ_O = E + S

CSSIOIEOCIECIEECCI

1111概率S_EIU_IQ_P = E + C

CSSIOIEOCIPIEPCI

1111概率S_EIU_IQ_M = C + S

MSIOIEOCIPIEPCI

1111 概 率 S_EIU_I_S = I

CSSSESSEESCSECSEECCS 1111概

率S_EIU_I_A = V + S

CSSSESSEESCSECSEECCS 1111概

率S_EIU_I_O = E + S

CSSSOSEOCSECSEECCS 1111概率

S_EIU_I_P = E + C

CSSSOSEOCSPSEPCS

1111概率S_EIU_I_M = C + S

MSSOSEOMPSEPM

1111概率S_EIU_Q_S = Q

CSSIESIEESCIECIEECCI 1111概率

S_EIU_Q_A = V + S

CSSIESIEESCIECIEECCI 1111概率

S_EIU_Q_O = E + S

CSSIOIEOCIECIEECCI

1111概率S_EIU_Q_P = E + C

CSSIOIEOCIPIEPCI

1111概率S_EIU_Q_M = C + S

MSIOIEOCIPIEPCI

1111概率S_112概率Increment DU

1111 概 率 S_E = D + U

IDIQIQIIUIQIEEIQIDIEIDIEIUIDIDI 1111 概

率 S_C = I + D

CIQIQIIUIQIEEIQCIECIEIUCCI 1111 概 率

S_EDU_IQ_S = I + Q

CSSIIUSIEESCIECIEIUCCI

1111概率S_EDU_IQ_A = V + S

CSSIIUSIEESCIECIEIUCCI 1111概率

S_EDU_IQ_O = E + S

CSSIIUSIEOCIECIEIUCCI 1111概率

S_EDU_IQ_P = E + C

CSSIIUSIEOCIPIEIUCCI

1111概率S_EDU_IQ_M = C + S

MSIIUSIEOCIPIEIUCCI

1111 概 率 S_EDU_I_S = I

CSSSSSSUSSSEESSCSECSESUCCS 1111 概

率 S_EDU_I_A = V + S

CSSSSSSUSSSEESSCSECSESUCCS 1111概

率S_EDU_I_O = E + S

CSSSSSSUSSSEOSCSECSOUCCS 1111 概

率 S_EDU_I_P = E + C

CSSSSSSUSSSEOSCSPSOUCCS 1111 概

率 S_EDU_I_M = C + S

MSSSSSUSSSEOSMPSOUCM 1111 概 率

S_EDU_Q_S = Q

CISISIIUISIEEISCIECIEIUCCI 1111 概 率

S_EDU_Q_A = V + S

CISISIIUISIEEISCIECIEIUCCI 1111 概 率

S_EDU_Q_O = E + S

CISISIIUISIEEISCIECIEIUCCI 1111 概 率

S_EDU_Q_P = E + C

CISISIIUISIEEISCIPIEIUCCI

1111概率S_EDU_Q_M = C + SCISISIIUISIEEISCIPIEIUCCI

1112概率S_I

1001

1111 概 率 S_I_11 降 元 S = I

IDQIIIUIIDUDUQIDIDUIDIDUIUIDIDI

1111概率S_I_11肱展 增元

1111 概 率 S_I_11V = U + Q

IDQIIIUIIDUDVIDIDUIDIDUIUIDIDI

1111概率S_I_111概率Increment IU

1111 概 率 S_I_E = I + U

IDQIIEIIEDVIDIEIDIEEIDIDI 1111

概 率 S_I_C = I + D

CQIIEIIEDVCIECIEECCI

1111概率S_I_EIU_IQ_S = I + QCQIIEIIEDVCIECIEECCI

1111概率S_I_EIU_IQ_A = V + SCQIIEIIEDVCIECIEECCI

1111概率S_I_EIU_IQ_O = E + SCQIIEIIEDVCIECIEECCI

1111概率S_I_EIU_IQ_P = E + CCQIIEIIEDVCIPIEPCI

1111概率S_I_EIU_IQ_M = C + SCQIIEIIEDVCIPIEPCI

1111 概 率 S_I_EIU_I_S = I

CSSSESSEDVCSECSEECCS 1111概率

S_I_EIU_I_A = V + S

CSSSESSEDVCSECSEECCS 1111概率

S_I_EIU_I_O = E + S

CSSSOSEDVCSECSEECCS 1111概率

S_I_EIU_I_P = E + C

CSSSOSEDVCSPSEPCS

1111概率S_I_EIU_I_M = C + S

MSSOSEDVMPSEPM

1111概率S_I_EIU_Q_S = QCSIIEIIEDVCIECIEECCI

1111概率S_I_EIU_Q_A = V + SCSIIEIIEDVCIECIEECCI

1111概率S_I_EIU_Q_O = E + SCSIIEIIEDVCIECIEECCI

1111概率S_I_EIU_Q_P = E + C

CSIIEIIEDVCIPIEPCI

1111概率S_I_EIU_Q_M = C + S

MIIEIIEDVCIPIEPCI

1111概率S_I_112概率Increment DU

1111 概 率 S_I_E = D + U

IDQIIIUIIEDVIDIEIDIEIUIDIDI 1111 概

率 S_I_C = I + D

CQIIIUIIEDVCIECIEIUCCI 1111概率

S_I_EDU_IQ_S = I + Q

CQIIIUIIEDVCIECIEIUCCI

1111概率S_I_EDU_IQ_A = V + SCQIIIUIIEDVCIECIEIUCCI

1111概率S_I_EDU_IQ_O = E + S

CQIIIUIIEDVCIECIEIUCCI 1111概率

S_I_EDU_IQ_P = E + C

CQIIIUIIEDVCIPIEIUCCI

1111概率S_I_EDU_IQ_M = C + SCQIIIUIIEDVCIPIEIUCCI

1111 概 率 S_I_EDU_I_S = I

CSSSSUSSEDVCSECSESUCCS 1111概

率S_I_EDU_I_A = V + S

CSSSSUSSEDVCSECSESUCCS 1111概

率S_I_EDU_I_O = E + S

CSSSSUSSEDVCSECSOUCCS 1111概

率S_I_EDU_I_P = E + C

CSSSSUSSEDVCSPSOUCCS 1111概率

S_I_EDU_I_M = C + S

MSSSUSSEDVMPSOUCM

1111 概 率 S_I_EDU_Q_S = Q

CSIIIUIIEDVCIECIEIUCCI 1111概率

S_I_EDU_Q_A = V + S

CSIIIUIIEDVCIECIEIUCCI

1111概率S_I_EDU_Q_O = E + S

CSIIIUIIEDVCIECIEIUCCI 1111概率

S_I_EDU_Q_P = E + C

CSIIIUIIEDVCIPIEIUCCI

1111概率S_I_EDU_Q_M = C + S

MIIIUIIEDVCIPIEIUCCI

1113概率S_Q

1111 概 率 S_Q_ 降 元 S = Q

IDQQIIUQIDUDUIIDIDUIDIDUIUIDIDI

1111概率S_Q_肱展 增元

1111 概 率 S_Q_V = U + Q

IDQQIIVIDUDUIIDIDUIDIDUIUIDIDI

1111概率S_Q_概率Increment IU

1111 概 率 S_Q_E = I + U

IDQQIIVIEEIIDIEIDIEEIDIDI 1111 概 率

S_Q_C = I + D CQQIIVIEEICIECIEECCI

1111概率S_Q_EIU_IQ_S = I + Q CQQIIVIEEICIECIEECCI

1111概率S_Q_EIU_IQ_A = V + S CQQIIVIEEICIECIEECCI

1111概率S_Q_EIU_IQ_O = E + S CQQIIVIEEICIECIEECCI

1111概率S_Q_EIU_IQ_P = E + C CQQIIVIEEICIPIEPCI

1111概率S_Q_EIU_IQ_M = C + S CQQIIVIEEICIPIEPCI

1111 概 率 S_Q_EIU_I_S = I

CSSSSVSEESCSECSEECCS 1111概率

S_Q_EIU_I_A = V + S

CSSSSAEESCSECSEECCS 1111概率

S_Q_EIU_I_O = E + S

CSSSSAEOCSECSEECCS

1111概率S_Q_EIU_I_P = E + C

CSSSSAEOCSPSEPCS

1111概率S_Q_EIU_I_M = C + S

MSSSAEOMPSEPM

1111概率S_Q_EIU_Q_S = QCSSIIVIEEICIECIEECCI

1111概率S_Q_EIU_Q_A = V + SCSSIIVIEEICIECIEECCI

1111概率S_Q_EIU_Q_O = E + SCSSIIVIEEICIECIEECCI

1111概率S_Q_EIU_Q_P = E + C

CSSIIVIEEICIPIEPCI

1111概率S_Q_EIU_Q_M = C + S

MSIIVIEEICIPIEPCI

1111概率S_Q_概率Increment DU

1111 概 率 S_Q_E = D + U

IDQQIIVIEEIIDIEIDIEIUIDIDI 1111 概

率 S_Q_C = I + D

CQQIIVIEEICIECIEIUCCI

1111概率S_Q_EDU_IQ_S = I + QCQQIIVIEEICIECIEIUCCI

1111概率S_Q_EDU_IQ_A = V + SCQQIIVIEEICIECIEIUCCI

1111概率S_Q_EDU_IQ_O = E + S

CQQIIVIEEICIECIEIUCCI

1111概率S_Q_EDU_IQ_P = E + C CQQIIVIEEICIPIEIUCCI

1111概率S_Q_EDU_IQ_M = C + S CQQIIVIEEICIPIEIUCCI

1111 概 率 S_Q_EDU_I_S = I

CSSSSVSEESCSECSESUCCS 1111概率

S_Q_EDU_I_A = V + S

CSSSSAEESCSECSESUCCS 1111概率

S_Q_EDU_I_O = E + S

CSSSSAEOCSECSOUCCS

1111概率S_Q_EDU_I_P = E + C

CSSSSAEOCSPSOUCCS

1111概率S_Q_EDU_I_M = C + S

MSSSAEOMPSOUCM

1111概率S_Q_EDU_Q_S = Q CSSIIVIEEICIECIEIUCCI

1111概率S_Q_EDU_Q_A = V + S CSSIIVIEEICIECIEIUCCI

1111概率S_Q_EDU_Q_O = E + S CSSIIVIEEICIECIEIUCCI

1111概率S_Q_EDU_Q_P = E + C

CSSIIVIEEICIPIEIUCCI

1111概率S_Q_EDU_Q_M = C + S

MSIIVIEEICIPIEIUCCI

2降元概率DU

21 降 元 E = D + U

CSSIDUSCUDUSCIIUCCUDUCCI 21降元C

= I + D

IDSSIDUSIDUDUSIDIIUIDIDUDUIDIDI 2111概率

S

2222 概 率 S_11 降 元 S = I + Q

IDIQIQIDUIQIDUDUIQIDIIUIDIDUDUIDIDI

2222概率S_11肽展 增元

2222 概 率 S_11V = U + Q

IDIQIQIDUIQIDUDUIQIDIIUIDIDUDUIDIDI

2222概率S_111概率Increment IU 2222 概

率 S_E = I + U

IDIQIQIEIQIEEIQIDIEIDIEEIDIDI 2222 概

率 S_C = I + D

CIQIQIEIQIEEIQCIECIEECCI 2222 概 率

S_EIU_IQ_S = I + Q

CSSIESIEESCIECIEECCI

2222概率S_EIU_IQ_A = V + S

CSSIESIEESCIECIEECCI 2222概率

S_EIU_IQ_O = E + S

CSSIOIEOCIECIEECCI

2222概率S_EIU_IQ_P = E + C

CSSIOIEOCIPIEPCI

2222概率S_EIU_IQ_M = C + S

MSIOIEOCIPIEPCI

2222 概 率 S_EIU_I_S = I

CSSSESSEESCSECSEECCS 2222概

率S_EIU_I_A = V + S

CSSSESSEESCSECSEECCS 2222概

率S_EIU_I_O = E + S

CSSSOSEOCSECSEECCS 2222概率

S_EIU_I_P = E + C

CSSSOSEOCSPSEPCS

2222概率S_EIU_I_M = C + S

MSSOSEOMPSEPM

2222 概 率 S_EIU_Q_S = Q

CSSIESIEESCIECIEECCI 2222概率

S_EIU_Q_A = V + S

CSSIESIEESCIECIEECCI 2222概率

S_EIU_Q_O = E + S

CSSIOIEOCIECIEECCI

2222概率S_EIU_Q_P = E + C

CSSIOIEOCIPIEPCI

2222概率S_EIU_Q_M = C + S

MSIOIEOCIPIEPCI

2222概率S_112概率Increment DU

2222 概 率 S_E = D + U

IDIQIQIEIQIEEIQIDIIUIDIEEIDIDI 2222 概

率 S_C = I + D

CIQIQIEIQIEEIQCIIUCIEECCI 2222 概 率

S_EDU_IQ_S = I + Q

CSSIESIEESCIIUCIEECCI

2222概率S_EDU_IQ_A = V + S

CSSIESIEESCIIUCIEECCI 2222概率

S_EDU_IQ_O = E + S

CSSIOIEOCIIUCIEECCI

2222概率S_EDU_IQ_P = E + C

CSSIOIEOCIIUCIEPCI

2222概率S_EDU_IQ_M = C + S

MSIOIEOCIIUCIEPCI

2222概率S_EDU_I_S = I

CSSSSSESSSEESSCSSUCSEECCS 2222 概

率 S_EDU_I_A = V + S

CSSSSSESSSEESSCSSUCSEECCS 2222 概

率 S_EDU_I_O = E + S

CSSSSSOSSEOSCSSUCSEECCS 2222 概

率 S_EDU_I_P = E + C

CSSSSSOSSEOSCSSUCSEPCS 2222 概 率

S_EDU_I_M = C + S

MSSSSOSSEOSMSUMEPM

2222 概 率 S_EDU_Q_S = Q

CISISIEISIEEISCIIUCIEECCI 2222概

率S_EDU_Q_A = V + S

CISISIEISIEEISCIIUCIEECCI 2222概

率S_EDU_Q_O = E + S

CISISIEISIEEISCIIUCIEECCI 2222概

率S_EDU_Q_P = E + C

CISISIEISIEEISCIIUCIEPCI 2222概率

S_EDU_Q_M = C + S

CISISIEISIEEISCIIUCIEPCI

2112概率S_I

10011111

2222 概 率 S_I_11 降 元 S = I

IDQQIDUQIDUDUQIDIIUIDIDUDUIDIDI

2222概率S_I_11肽展 增元

2222 概 率 S_I_11V = U + Q

IDQQIDVIDUDVIDIIUIDIDUDUIDIDI

2222概率S_I_111概率Increment IU

2222 概 率 S_I_E = I + U

IDQQIDVIEDVIDIEIDIEEIDIDI

2222 概 率 S_I_C = I + D

CQQCVIEDVCIECIEECCI 2222概率

S_I_EIU_IQ_S = I + Q

CQQCVIEDVCIECIEECCI

2222概率S_I_EIU_IQ_A = V + S CQQCVIEDVCIECIEECCI

2222概率S_I_EIU_IQ_O = E + S

CQQCVIEDVCIECIEECCI 2222概率

S_I_EIU_IQ_P = E + C

CQQCVIEDVCIPIEPCI

2222概率S_I_EIU_IQ_M = C + S CQQCVIEDVCIPIEPCI

2222 概 率 S_I_EIU_I_S = I

CSSCVSEDVCSECSEECCS 2222概率

S_I_EIU_I_A = V + S

CSSCAEDVCSECSEECCS 2222概率

S_I_EIU_I_O = E + S

CSSCAEDVCSECSEECCS 2222概率

S_I_EIU_I_P = E + C

CSSCAEDVCSPSEPCS

2222概率S_I_EIU_I_M = C + S

MSCAEDVMPSEPM

2222概率S_I_EIU_Q_S = Q CSSCVIEDVCIECIEECCI

2222概率S_I_EIU_Q_A = V + S CSSCVIEDVCIECIEECCI

2222概率S_I_EIU_Q_O = E + S

CSSCVIEDVCIECIEECCI 2222概率

S_I_EIU_Q_P = E + C

CSSCVIEDVCIPIEPCI

2222概率S_I_EIU_Q_M = C + S

MSCVIEDVCIPIEPCI

2222概率S_I_112概率Increment DU 2222 概

率 S_I_E = D + U

IDQQIDVIEDVIDIIUIDIEEIDIDI 2222 概 率

S_I_C = I + D CQQCVIEDVCIIUCIEECCI

2222概率S_I_EDU_IQ_S = I + Q CQQCVIEDVCIIUCIEECCI

2222概率S_I_EDU_IQ_A = V + S CQQCVIEDVCIIUCIEECCI

2222概率S_I_EDU_IQ_O = E + S

CQQCVIEDVCIIUCIEECCI 2222概率

S_I_EDU_IQ_P = E + C

CQQCVIEDVCIIUCIEPCI

2222概率S_I_EDU_IQ_M = C + S CQQCVIEDVCIIUCIEPCI

2222 概 率 S_I_EDU_I_S = I

CSSCVSEDVCSSUCSEECCS 2222概率

S_I_EDU_I_A = V + S

CSSCAEDVCSSUCSEECCS 2222概率

S_I_EDU_I_O = E + S

CSSCAEDVCSSUCSEECCS 2222概率

S_I_EDU_I_P = E + C

CSSCAEDVCSSUCSEPCS 2222概率

S_I_EDU_I_M = C + S

MSCAEDVMSUMEPM

2222概率S_I_EDU_Q_S = Q

CSSCVIEDVCIIUCIEECCI

2222概率S_I_EDU_Q_A = V + S

CSSCVIEDVCIIUCIEECCI 2222概率

S_I_EDU_Q_O = E + S

CSSCVIEDVCIIUCIEECCI 2222概率

S_I_EDU_Q_P = E + C

CSSCVIEDVCIIUCIEPCI

2222概率S_I_EDU_Q_M = C + S

MSCVIEDVCIIUCIEPCI

2113概率S_Q

2222 概 率 S_Q_ 降 元 S = Q

IDQQIDUQIDUDUIIDIIUIDIDUDUIDIDI

2222概率S_Q_肱展 增元

2222 概 率 S_Q_V = U + Q

IDQQIDVIDUDUIIDIIUIDIDUDUIDIDI

2222概率S_Q_概率Increment IU 2222 概

率 S_Q_E = I + U

IDQQIDVIEEIIDIEIDIEEIDIDI 2222 概

率 S_Q_C = I + D

CQQCVIEEICIECIEECCI

2222概率S_Q_EIU_IQ_S = I + Q CQQCVIEEICIECIEECCI

2222概率S_Q_EIU_IQ_A = V + S CQQCVIEEICIECIEECCI

2222概率S_Q_EIU_IQ_O = E + S CQQCVIEEICIECIEECCI

2222概率S_Q_EIU_IQ_P = E + C CQQCVIEEICIPIEPCI

2222概率S_Q_EIU_IQ_M = C + S CQQCVIEEICIPIEPCI

2222概率S_Q_EIU_I_S = I

CSSCVSEESCSECSEECCS

2222概率S_Q_EIU_I_A = V + S

CSSCAEESCSECSEECCS

2222概率S_Q_EIU_I_O = E + S

CSSCAEOCSECSEECCS

2222概率S_Q_EIU_I_P = E + C

CSSCAEOCSPSEPCS

2222概率S_Q_EIU_I_M = C + S

MSCAEOMPSEPM

2222概率S_Q_EIU_Q_S = QCSSCVIEEICIECIEECCI

2222概率S_Q_EIU_Q_A = V + SCSSCVIEEICIECIEECCI

2222概率S_Q_EIU_Q_O = E + SCSSCVIEEICIECIEECCI

2222概率S_Q_EIU_Q_P = E + C

CSSCVIEEICIPIEPCI

2222概率S_Q_EIU_Q_M = C + S

MSCVIEEICIPIEPCI

2222概率S_Q_概率Increment DU 2222 概

率 S_Q_E = D + U

IDQQIDVIEEIIDIIUIDIEEIDIDI 2222 概

率 S_Q_C = I + D

CQQCVIEEICIIUCIEECCI

2222概率S_Q_EDU_IQ_S = I + QCQQCVIEEICIIUCIEECCI

2222概率S_Q_EDU_IQ_A = V + SCQQCVIEEICIIUCIEECCI

2222概率S_Q_EDU_IQ_O = E + SCQQCVIEEICIIUCIEECCI

2222概率S_Q_EDU_IQ_P = E + C

CQQCVIEEICIIUCIEPCI

2222概率S_Q_EDU_IQ_M = C + SCQQCVIEEICIIUCIEPCI

2222 概 率 S_Q_EDU_I_S = I

CSSCVSEESCSSUCSEECCS 2222概率

S_Q_EDU_I_A = V + S

CSSCAEESCSSUCSEECCS 2222概率

S_Q_EDU_I_O = E + S

CSSCAEOCSSUCSEECCS

2222概率S_Q_EDU_I_P = E + C

CSSCAEOCSSUCSEPCS

2222概率S_Q_EDU_I_M = C + S

MSCAEOMSUMEPM

2222概率S_Q_EDU_Q_S = QCSSCVIEEICIIUCIEECCI

2222概率S_Q_EDU_Q_A = V + SCSSCVIEEICIIUCIEECCI

2222概率S_Q_EDU_Q_O = E + SCSSCVIEEICIIUCIEECCI

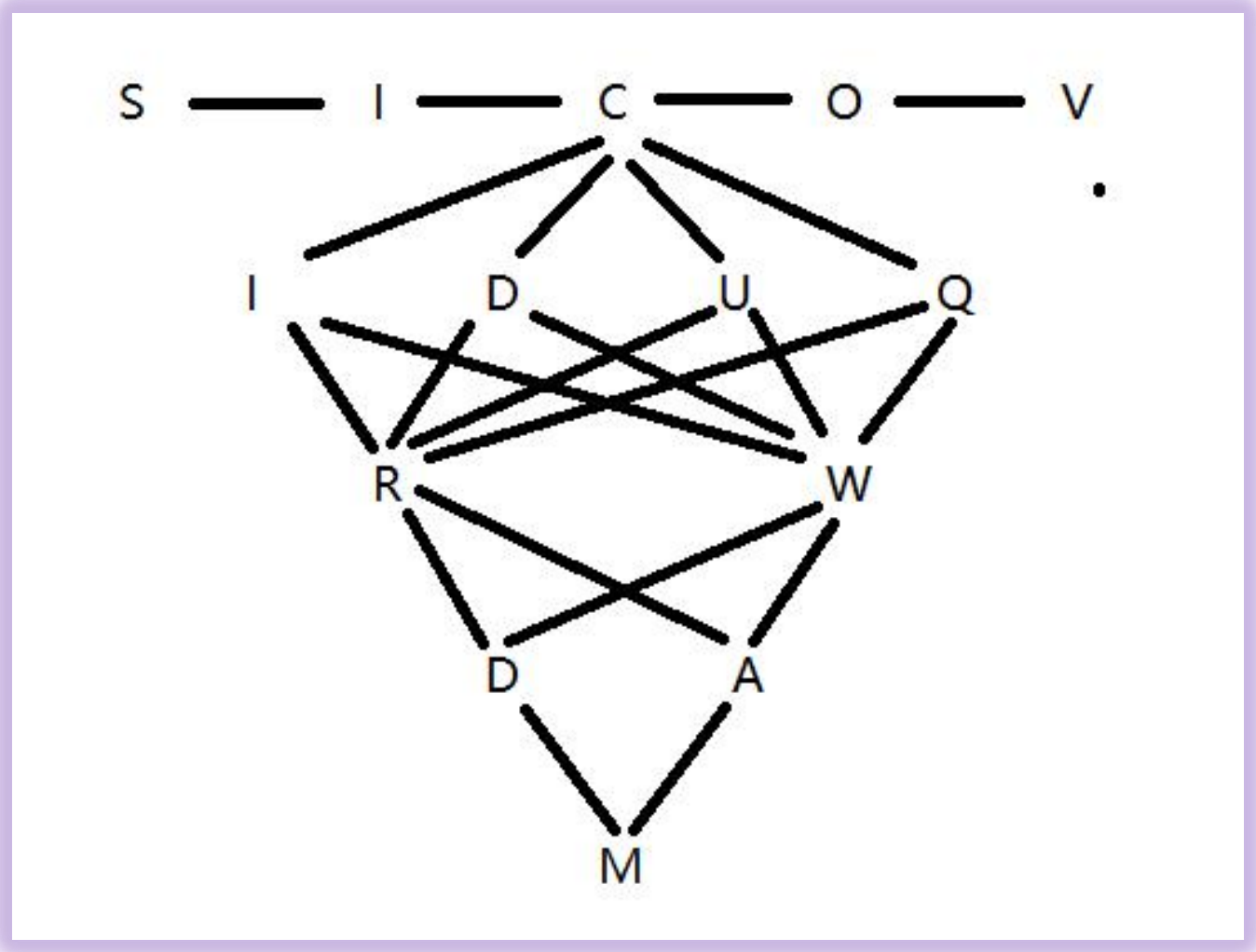
2222概率S_Q_EDU_Q_P = E + C

CSSCVIEEICIIUCIEPCI

2222概率S_Q_EDU_Q_M = C + S

MSCVIEEICIIUCIEPCI

通过元基的肽展公式, 我设计了两种模式, 一种的是增元模式, 一种是降元模式, 将肽化好的元基表进行比值变换, 这里有很多种变换算法, 变换好后, 我开始用肽展降元和增元公式函数进行变换,生成的 TI 概率钥匙用于数据识别. 这个识别的元基串用于服务器端计算. TD 钥匙则用于数据验证. 用于复检.



在数据的SICOV 模式流动下, 将控制层进行元基编码和解码进行拆分, 然后读写分离, 分离后进行函数化在映射和直接寻址间, 然后集成在管理系统中. 我命名为元基加密组件. 我准备分离出来写一个加密解密算法 API, 用于各种不同的加密场景. 我解释下这些字母:

S 数据I 输入C 控制O 输出V 观测I 增加D 减少U 改变Q 查看R 读W 写D 直接寻址A 计算内存

M 管理器

```
//函数实现
public static void main(String[] argv) {
        FullDNATokenPDI pDE_RNA_FullFormular= new FullDNATokenPDI();
        @SuppressWarnings("unused")
        String initonKeys= "EIU/0. 6/EDU/0. 4/si/0. 3/sq/0. 7/EIU/0. 5/EDU/0. 5/si/0. 632/sq/0. 368";
        pDE_RNA_FullFormular. key[0]= 0. 6;
        pDE_RNA_FullFormular. key[1]= 0. 3;
        pDE_RNA_FullFormular. key[2]= 0. 5;
        pDE_RNA_FullFormular. key[3]= 0. 632;
        pDE_RNA_FullFormular. text= "控制吸收";
        pDE_RNA_FullFormular. pdw= pDE_RNA_FullFormular. initonSect(pDE_RNA_FullFormular.
text);
        System. out. println("原文: " + pDE_RNA_FullFormular. text);
        //pDE_RNA_FullFormular. pdw= "字典保密: MSIOCUOCIPCUPCI"; String[] lock=
        new String[12];
  lock[0] = "A"; lock[3] = "O"; lock[6] = "P"; lock[9] = "M";
  lock[1] = "V"; lock[4] = "E"; lock[7] = "C"; lock[10] = "S";
  lock[2] = "I"; lock[5] = "D"; lock[8] = "U"; lock[11] = "Q"; int i=
  (int)(Math. random()* 12)% 12; pDE_RNA_FullFormular. lock+= lock[i];
  i= (int)(Math. random()* 12)% 12;
  pDE_RNA_FullFormular. lock+= lock[i]; i=
  (int)(Math. random()* 12)% 12;
  pDE_RNA_FullFormular. lock+= lock[i]; i=
  (int)(Math. random()* 12)% 12;
  pDE_RNA_FullFormular. lock+= lock[i];
        for(i= 0; i< pDE_RNA_FullFormular. pdw. length(); i++) {
            pDE_RNA_FullFormular. code+= pDE_RNA_FullFormular. lock + pDE_RNA_FullFormular.
```

```
pdw. charAt(i);
            }
            System. out. println(" 肽 语 : "+ pDE_RNA_FullFormular. pdw); System. out.
            println(" 肽 锁 : "+ pDE_RNA_FullFormular. lock); System. out. println("散列肽语:
            "+ pDE_RNA_FullFormular. code); pDE_RNA_FullFormular. bys= "0. 6/0. 3/0. 5/0.
            632";
            System. out. println("静态密钥: "+ pDE_RNA_FullFormular. bys);
            pDE_RNA_FullFormular. doKeyPress(pDE_RNA_FullFormular. code, pDE_RNA_FullFormular,
false);
            System. out. println("静态肽展降元概率钥匙E: "+ pDE_RNA_FullFormular. pdedeKey); System. out. println("
            静态肽展降元概率钥匙S: "+ pDE_RNA_FullFormular. pdedsKey); System. out. println("静态肽展降元: "+
            pDE_RNA_FullFormular. pds);

            System. out. println("静态肽展增元概率钥匙E: "+ pDE_RNA_FullFormular. pdeieKey); System. out. println("
            静态肽展增元概率钥匙S: "+ pDE_RNA_FullFormular. pdeisKey); System. out. println("静态肽展增元: "+
            pDE_RNA_FullFormular. pde);

            pDE_RNA_FullFormular. time= "" + System. currentTimeMillis(); pDE_RNA_FullFormular. cacheId=
            "ID" + Math. random() + ": " + Math. random(); System. out. println("时间: " + pDE_RNA_FullFormular.
            time);
            System. out. println("账号随机缓存字符串: " + pDE_RNA_FullFormular. cacheId);
            pDE_RNA_FullFormular. session_key= pDE_RNA_FullFormular. pde; System. out.
            println("Session: " + pDE_RNA_FullFormular. session_key); System. out.
println("==========================================================================
======");
            System. out. println("开始前序验证: ");
            System. out. println("开始Session解析: " + pDE_RNA_FullFormular. session_key); System. out. println("
            开始概率钥匙解析: " + pDE_RNA_FullFormular. pdedeKey+
pDE_RNA_FullFormular. pdedsKey
```

```
+ pDE_RNA_FullFormular. pdeieKey+ pDE_RNA_FullFormular. pdeisKey);

        FullDNATokenPDI pDE_RNA_FullFormular1= new FullDNATokenPDI(); pDE_RNA_FullFormular1.
        pdedeKey= pDE_RNA_FullFormular. pdedeKey. toString(); pDE_RNA_FullFormular1. pdedsKey=
        pDE_RNA_FullFormular. pdedsKey. toString(); pDE_RNA_FullFormular1. pdeieKey=
        pDE_RNA_FullFormular. pdeieKey. toString(); pDE_RNA_FullFormular1. pdeisKey=
        pDE_RNA_FullFormular. pdeisKey. toString();

        pDE_RNA_FullFormular. doKeyUnPress(pDE_RNA_FullFormular. code,
pDE_RNA_FullFormular1, true);
        System. out. println();
        System. out. println("得到原降元元基DNA序列: "+ pDE_RNA_FullFormular. pds); System. out. println("
        得到新降元元基DNA序列: "+ pDE_RNA_FullFormular1. pds); System. out. println("得到原元基DNA序
        列: "+ pDE_RNA_FullFormular. pde); System. out. println("得到新元基DNA序列: "+
        pDE_RNA_FullFormular1. pde); System. out. println("验证正确？ ");
        System. out. println(pDE_RNA_FullFormular. pde. equals(pDE_RNA_FullFormular1. pde)? "正确": "失败");

        System. out. println("========================================================================== ");
        System. out. println("开始后序验证: ");
        FullDNATokenPDI pDE_RNA_FullFormular2= new FullDNATokenPDI(); pDE_RNA_FullFormular2.
        pdeieKey= pDE_RNA_FullFormular. pdedeKey. toString(); pDE_RNA_FullFormular2. pdeisKey=
        pDE_RNA_FullFormular. pdedsKey. toString(); pDE_RNA_FullFormular2. pdedeKey=
        pDE_RNA_FullFormular. pdeieKey. toString(); pDE_RNA_FullFormular2. pdedsKey=
        pDE_RNA_FullFormular. pdeisKey. toString(); System. out. println("准备计算元基DNA序列: "+
        pDE_RNA_FullFormular1. pde); pDE_RNA_FullFormular2.
        doSessionKeyUnPress(pDE_RNA_FullFormular1. pde,
```

```
pDE_RNA_FullFormular2, true); System.
        out. println();
        System. out. println("得到原续降元元基DNA序列: "+ pDE_RNA_FullFormular1. pds); System. out. println("
        得到后续降元元基DNA序列: "+ pDE_RNA_FullFormular2. pds); System. out. println("验证正确？ ");
        System. out. println(pDE_RNA_FullFormular1. pds. equals(pDE_RNA_FullFormular2. pds)? "正确": "失败");

        System. out. println("==========================================================================
==");
        System. out. println("开始整序验证: ");
        FullDNATokenPDI pDE_RNA_FullFormular3= new FullDNATokenPDI();

        pDE_RNA_FullFormular3.  pdeieKey=  pDE_RNA_FullFormular.  pdeieKey.  toString();
        pDE_RNA_FullFormular3.  pdeisKey=  pDE_RNA_FullFormular.  pdeisKey.  toString();
        pDE_RNA_FullFormular3.  pdedeKey=  pDE_RNA_FullFormular.  pdeieKey.  toString();
        pDE_RNA_FullFormular3. pdedsKey= pDE_RNA_FullFormular. pdeisKey. toString();

        System.  out.  println("准 备 计 算 元 基 DNA 序 列 ： "+  pDE_RNA_FullFormular1.  pde);
        pDE_RNA_FullFormular3. doFullSessionKeyUnPress(pDE_RNA_FullFormular1. pde,
pDE_RNA_FullFormular3, true); System.
        out. println();
        System. out. println("得到原续降元元基DNA序列: "+ pDE_RNA_FullFormular1. pds); System. out. println("
        得到后续降元元基DNA序列: "+ pDE_RNA_FullFormular3. pds); System. out. println("验证正确？ ");
        System. out. println(pDE_RNA_FullFormular1. pds. equals(pDE_RNA_FullFormular3. pds)? "正确": "失败");
```

```
        System. out. println("准备整序计算元基DNA序列: "+ pDE_RNA_FullFormular1. pde); System. out. println("
        准备整序计算元基DNA序列: "+ pDE_RNA_FullFormular3. pde);
        System. out. println(pDE_RNA_FullFormular1. pde. equals(pDE_RNA_FullFormular3. pde)? "正确": "失败");


    }
    private void do_PDE_RNA_FullFormular_FullBack(Initon initon, FullDNATokenPDI
pDE_RNA_FullFormular3
            , boolean bYS)
        { Initon InitonPDE=
        initon;
        InitonLinkDNA initonLinkDNA= new InitonLinkDNA();
        Initon InitonPDE1V= doIncrementV(InitonPDE, initonLinkDNA, pDE_RNA_FullFormular3);
        Initon InitonPDE1E= doIncrementE(InitonPDE1V, initonLinkDNA, pDE_RNA_FullFormular3, bYS);
        Initon InitonPDE1C= doIncrementC(InitonPDE1E, initonLinkDNA, pDE_RNA_FullFormular3);
        Initon InitonPDE1S= doIncrementS(InitonPDE1C, initonLinkDNA, pDE_RNA_FullFormular3, bYS);

        Initon InitonPDE1A= doIncrementA(InitonPDE1S, initonLinkDNA, pDE_RNA_FullFormular3);
        Initon InitonPDE1O= doIncrementO(InitonPDE1A, initonLinkDNA, pDE_RNA_FullFormular3);
        Initon InitonPDE1P= doIncrementP(InitonPDE1O, initonLinkDNA, pDE_RNA_FullFormular3);
        Initon InitonPDE2= doIncrementM(InitonPDE1P, initonLinkDNA, pDE_RNA_FullFormular3);
        while(InitonPDE2. hasNext()) {
            pDE_RNA_FullFormular3. pde+= InitonPDE2. getStore();
            InitonPDE2= InitonPDE2. next;
        }
        pDE_RNA_FullFormular3. pde+= InitonPDE2. getStore();
        while(InitonPDE2. hasPrev())
            { InitonPDE2= InitonPDE2.
            prev;
        }

        Initon InitonPDEM= doDecrementM(InitonPDE2, initonLinkDNA, pDE_RNA_FullFormular3);
```

```
        Initon InitonPDEP= doDecrementP(InitonPDEM, initonLinkDNA, pDE_RNA_FullFormular3);
        Initon InitonPDEO= doDecrementO(InitonPDEP, initonLinkDNA, pDE_RNA_FullFormular3);
        Initon InitonPDEA= doDecrementA(InitonPDEO, initonLinkDNA, pDE_RNA_FullFormular3);

        Initon InitonPDES= doDecrementS(InitonPDEA, initonLinkDNA, pDE_RNA_FullFormular3, bYS);
        Initon InitonPDEC= doDecrementC(InitonPDES, initonLinkDNA, pDE_RNA_FullFormular3);
        Initon InitonPDEE= doDecrementE(InitonPDEC, initonLinkDNA, pDE_RNA_FullFormular3, bYS);
        Initon InitonPDE1= doDecrementV(InitonPDEE, initonLinkDNA, pDE_RNA_FullFormular3);
        while(InitonPDE1. hasNext()) {
            pDE_RNA_FullFormular3. pds+= InitonPDE1. getStore();
            InitonPDE1= InitonPDE1. next;
        }
        pDE_RNA_FullFormular3. pds+= InitonPDE1. getStore();
        while(InitonPDE1. hasPrev())
            { InitonPDE1= InitonPDE1.
            prev;
        }
    }
//下面 9 页 plsql 的 orm  语法我也并入第二卷一开始是为了追梁壁荧写的, 追不到就算了.


package org. tinos. language. pletl; import
java. io. File;
import java. util. HashMap; import javax.
swing. JTextPane;
import OSI. OSU. MSQ. sets. stable. StableData; import OSI. OSU.
OEI. PVI. document. load. LoadFile;
import OSI. OSU. OVU. MVQ. GUI. nodeView. NodeShow; import OSI.
OSU. OVU. MVU. GUI. nodeEdit. LinkList; import OSI. OSU. OVU. MVU.
GUI. nodeEdit. LinkNode;
import OSI. OSU. PSQ. OEU. document. neroCell. BootNeroCell; public class
PLETLImpl implements PLETLIntef{
```

```
    @Override
    public boolean doNeroFlow(JTextPane rightBotJTextPane, NodeShow nodeView, LinkList first
            , String documentFlowAddress, HashMap<String, Object> inputMap) {
        //很好的将 《德塔 socket plsql 数据库》 和 《德塔 ETL》变成脑的记忆和计算中枢配合.
        //别急, 这个组合虽然没有自主意识, 但是已经形成了 VPCS 计算神经元的单株 锥形.
20200322 罗瑶光
        try {
            String fileCurrentpath= documentFlowAddress; File file=
            new File(fileCurrentpath);
            if(!file. isFile()) {
                System. out. println(StableData. ATTENSION_RECHOICE); return false;
            }
            LinkNode needDeleteNode= first. first;
            while(needDeleteNode!= null) {
                first. first= first. deletNode(first. first, needDeleteNode. name
                        , needDeleteNode. ID, needDeleteNode. primaryKey); if(null==
                needDeleteNode. next) {
                    break;
                }
                needDeleteNode= needDeleteNode. next;
            }
            first. first= LoadFile. Load(first. first, nodeView, file, first); BootNeroCell.
            bootCell(first. first, rightBotJTextPane, null);
        } catch(Exception loadE) { loadE.
            printStackTrace();
        }
        return true;
    }
    //大爱如此
```

```
}

//接口
package org. tinos. language. plorm;
import java. util. Map;
public interface PLORMInterf{
    public String getPLSQL();
    public void setPLSQL(String pLSQL);
    public PLORMInterf withTableCreate(String tableName); public
    PLORMInterf withTableDelete(String tableName); public PLORMInterf
    withTableInsert(String tableName); public PLORMInterf
    withTableUpdate(String tableName); public PLORMInterf
    withTableSelect(String tableName); public PLORMInterf getCulumns();
    public PLORMInterf startAtRootDir(String rootAddress); public
    PLORMInterf withBaseName(String baseName); public PLORMInterf
    withCondition(String conditionType); public PLORMInterf let(String
    leftSet);
    public PLORMInterf lessThanAndEqualTo(String compareSet);
    public PLORMInterf equalTo(String compareSet); public
    PLORMInterf lessThan(String compareSet); public PLORMInterf
    greatThan(String compareSet);
    public PLORMInterf greatThanAndEqualTo(String compareSet);
    public PLORMInterf notEqualTo(String compareSet) ;
    public PLORMInterf in(String compareSet) ; public
    PLORMInterf notIn(String compareSet) ; public
    PLORMInterf equals(String compareSet);
    public PLORMInterf notEquals(String compareSet);
    public PLORMInterf innerJoinWithTable(String baseName, String tableName);
    public PLORMInterf withRelation(String relationType) ;
```

```
    public PLORMInterf as(String compareSet) ;
    public PLORMInterf upTo(String compareSet);
    public PLORMInterf withAggregation(String aggregationType);
    public PLORMInterf changeCulumnName(String newCulumnName, String oldCulumnName);
    public PLORMInterf withCulumnName(String culumnName, String dataType); public PLORMInterf
    withCulumnValue(String culumnName, String culumnValue); public PLORMInterf checkErrors(String
    string);
    public PLORMInterf fixErrors(String string);
    public PLORMInterf finalExec(boolean b) throws Exception;
    public Map<String, Object> returnAsMap();
    public PLORMInterf checkAndFixPlsqlGrammarErrors();
    public PLORMInterf checkAndFixSystemEnvironmentErrors();
}

//描述
package org. tinos. language. plorm; import
java. util. Map;

import org. ME. plsql. db. plsql. imp. ExecPLSQLImp; public class
PLORMImpl implements PLORMInterf{
    private String PLSQL= ""; private
    String[] PLSQLArray;
    private Map<String, Object> map; public
    String getPLSQL() {
        return PLSQL;
    }

    public void setPLSQL(String pLSQL) { PLSQL=
        pLSQL;
    }
```

```
public PLORMImpl startAtRootDir(String rootAddress) { PLSQL= Const.
        SET_ROOT+ Const. COLON+ rootAddress
                + Const. SEMICOLON;
        return this;
}
public PLORMImpl withBaseName(String baseName) {
        PLSQL+= Const. SEMICOLON+ Const. BASE_NAME+ Const. COLON
                + baseName;
        return this;
}
public PLORMImpl withTableSelect (String tableName) {
        PLSQL+= Const. SEMICOLON+ Const. TABLE_NAME+ Const. COLON
                + tableName
                + Const. COLON+ Const. SELECT; return
        this;
}
public PLORMImpl withTableCreate(String tableName) {
        PLSQL+= Const. SEMICOLON+ Const. TABLE_NAME+ Const. COLON
                + tableName
                + Const. COLON+ Const. CREATE; return this;
}
public PLORMImpl withTableDelete(String tableName) {
        PLSQL+= Const. SEMICOLON+ Const. TABLE_NAME+ Const. COLON
                + tableName
                + Const. COLON+ Const. DELETE; return
        this;
}
public PLORMImpl withTableInsert(String tableName) {
```

```
        PLSQL+= Const. SEMICOLON+ Const. TABLE_NAME+ Const. COLON
                + tableName
                + Const. COLON+ Const. INSERT; return this;
    }
    public PLORMImpl withTableUpdate(String tableName) {
        PLSQL+= Const. SEMICOLON+ Const. TABLE_NAME+ Const. COLON
                + tableName
                + Const. COLON+ Const. UPDATE; return this;
    }
    public PLORMImpl withCondition(String conditionType) {
        PLSQL+= Const. SEMICOLON+ Const. CONDITION+ Const. COLON
                + conditionType;
        return this;
    }
    public PLORMImpl let(String leftSet)
        { PLSQL+= Const. COLON+ leftSet; return
        this;
    }
    public PLORMImpl lessThanAndEqualTo(String compareSet) { PLSQL+= Const.
        LESS_THAN_AND_EQUAL_TO+ compareSet; return this;
    }
    public PLORMImpl equalTo(String compareSet) { PLSQL+=
        Const. EQUAL_TO+ compareSet; return this;
    }
    public PLORMImpl lessThan(String compareSet) { PLSQL+=
        Const. LESS_THAN+ compareSet;
```

```
        return this;
    }
    public PLORMImpl greatThan(String compareSet) { PLSQL+=
        Const. GREAT_THAN+ compareSet; return this;
    }
    public PLORMImpl greatThanAndEqualTo(String compareSet) { PLSQL+= Const.
        GREAT_THAN_AND_EQUAL_TO+compareSet; return this;
    }
    public PLORMImpl notEqualTo(String compareSet) { PLSQL+=
        Const. NOT_EQUAL_TO+ compareSet; return this;
    }
    public PLORMImpl in(String compareSet) { PLSQL+=
        Const. IN+ compareSet; return this;
    }
    public PLORMImpl notIn(String compareSet) { PLSQL+=
        Const. NOT_IN+ compareSet; return this;
    }
    public PLORMImpl equals(String compareSet) { PLSQL+=
        Const. EQUALS+ compareSet; return this;
    }
    public PLORMImpl notEquals(String compareSet) { PLSQL+=
        Const. NOT_EQUALS+ compareSet; return this;
    }
```

```
public PLORMImpl innerJoinWithTable(String baseName, String tableName) { PLSQL+= Const.
    SEMICOLON+ Const. JOIN+ Const. COLON+ baseName
            + Const. COLON+ tableName; return
    this;
}
public PLORMImpl withRelation(String relationType) {
    PLSQL+= Const. SEMICOLON+ Const. RELATION+ Const. COLON
            + relationType; return
    this;
}
public PLORMImpl as(String compareSet) { PLSQL+=
    Const. AS+ compareSet; return this;
}
public PLORMImpl upTo(String compareSet) { PLSQL+=
    Const. UP_TO+ compareSet; return this;
}
public PLORMImpl withAggregation(String aggregationType) { PLSQL+= Const.
    SEMICOLON+ Const. WITH_AGGREGATION
            + Const. COLON+ aggregationType; return this;
}
public PLORMImpl getCulumns() {
    PLSQL+= Const. SEMICOLON+ Const. GET_CULUMNS;
    return this;
}
public PLORMImpl changeCulumnName(String newCulumnName, String oldCulumnName) { PLSQL+= Const.
    SEMICOLON+ Const. CHANGES_CULUMN_NAME+ Const. COLON
            + newCulumnName+ Const. COLON+ oldCulumnName;
```

```
        return this;
    }
    public PLORMImpl withCulumnName(String culumnName, String dataType) {
        PLSQL+= Const. SEMICOLON+ Const. CULUMN_NAME+ Const. COLON+ culumnName
                + Const. COLON+ dataType; return this;
    }
    public PLORMImpl withCulumnValue(String culumnName, String culumnValue) {
        PLSQL+= Const. SEMICOLON+ Const. CULUMN_VALUE+ Const. COLON+ culumnName
                + Const. COLON+ culumnValue; return
        this;
    }
    public PLORMInterf exec(boolean b) throws Exception { map=
        ExecPLSQLImp. ExecPLORM(this, true); return this;
    }

    @Override
    public PLORMInterf checkErrors(String string) { return this;
    }
    @Override
    public PLORMInterf fixErrors(String string) { return this;
    }

    @Override
    public PLORMInterf finalExec(boolean b) throws Exception { map=
        ExecPLSQLImp. ExecPLORM(this, true);
        return this;
```

```
}

@Override
public Map<String, Object> returnAsMap() { return this.
    map;
}

@Override
public PLORMInterf checkAndFixPlsqlGrammarErrors() {
    //string to array
    this. PLSQLArray= PLSQL. split(Const. SEMICOLON);
    //条件检查  1 过滤  2 修改  3 语义检测
    //1
    for(int i= 1; i< PLSQLArray. length; i++) {
        //1. 1  过滤相同句型
        //1. 2  过滤无效字符
        //1. 3  过滤攻击代码
        if(PLSQLArray[i]. equalsIgnoreCase(PLSQLArray[i- 1]))
            { PLSQLArray[i]= "";
        }
        PLSQLArray[i]= PLSQLArray[i]. replaceAll(">+", ">"); PLSQLArray[i]=
        PLSQLArray[i]. replaceAll("<+", "<"); PLSQLArray[i]= PLSQLArray[i].
        replaceAll("\\!+", "!"); PLSQLArray[i]= PLSQLArray[i]. replaceAll("\\~+",
        "~"); PLSQLArray[i]= PLSQLArray[i]. replaceAll("\\@+", "@");
        PLSQLArray[i]= PLSQLArray[i]. replaceAll("\\&&+", "&&");
        PLSQLArray[i]= PLSQLArray[i]. replaceAll("\\|\\|+", "||"); PLSQLArray[i]=
        PLSQLArray[i]. replaceAll("\\[+", "["); PLSQLArray[i]= PLSQLArray[i].
        replaceAll("\\]+", "]"); PLSQLArray[i]= PLSQLArray[i]. replaceAll("\\: +", ":
        ");
```

```
                PLSQLArray[i]= PLSQLArray[i]. replaceAll("\\s+", "");
            }
            //2
            //2. 1  修改错误比较符号
            //2. 2  修改错误语法关键字
            //2. 3  修改错误标注符号
            //3
            //3. 1  检测是否有关键字前后句段混乱
            //3. 2  检测是否有关键字  格式  倒置
            //3. 3  检测是否有关键字  句型  倒置
            //rerturn
            String string= "";
            for(int i= 0; i< PLSQLArray. length; i++) {
                string+= PLSQLArray[i]+ Const. SEMICOLON;
            }
            PLSQL= string;
            return this;
        }
        @Override
        public PLORMInterf checkAndFixSystemEnvironmentErrors() { return this;
        }
}
```

```
//常量
package org. tinos. language. plorm;
public class Const{
    public final static String SET_ROOT= "setRoot";
    public final static String BASE_NAME= "baseName";
    public final static String TABLE_NAME= "tableName";
    public final static String SELECT= "select";
    public final static String CREATE= "create";
    public final static String DELETE= "delete";
    public final static String INSERT= "insert";
    public final static String UPDATE= "update";
    public final static String CONDITION= "condition";
    public final static String LESS_THAN_AND_EQUAL_TO= "|<=|";
    public final static String EQUAL_TO= "|==|";
    public final static String LESS_THAN= "|<|";
    public final static String GREAT_THAN= "|>|";
    public final static String GREAT_THAN_AND_EQUAL_TO= "|>=|";
    public final static String NOT_EQUAL_TO= "|<>|";
    public final static String IN= "|in|";
    public final static String NOT_IN= "|!in|";
    public final static String EQUALS= "|equal|";
    public final static String NOT_EQUALS= "|!equal|";
    public final static String JOIN= "join";
    public final static String RELATION= "relation";
    public final static String AS= "|as|";
    public final static String UP_TO= "|~|";
    public final static String WITH_AGGREGATION= "aggregation";
    public final static String GET_CULUMNS= "getCulumns";
    public final static String CHANGES_CULUMN_NAME= "changeCulumnName";
    public final static String CULUMN_NAME= "culumnName";
```

```
    public final static String CULUMN_VALUE= "columnValue";
    public final static String SEMICOLON= "; ";
    public final static String COLON= ": ";
}

//ORM 示例  create
package org. tinos. language. plorm;
public class Create{
//动机: 准备将下面的plsql翻译成orm,  省的养疗经[17]的query模式太固定, 上手修改麻烦.
//    String plsql= "setRoot: C: /DetaDB; ";
//      plsql+= "baseName: ZYY; ";
//      plsql+= "tableName: zybc: create; " +
//                  "columnName: pk: ID: string; " +
//                  "columnName: uk:  打分: string; " +
//                  "columnName: uk:  中药名称: string; " +
//                  "columnName: uk:  笔记原文: string; " +
//                  "columnName: uk:  功效: string; " +
//                  "columnName: uk:  风险规避: string; " +
//                  "columnName: uk:  用量: string; " +
//                  "columnName: uk:  性味: string; " +
//                  "columnName: uk:  经脉: string; " +
//                  "columnName: uk:  中医馆药理: string; " +
//                  "columnName: uk:  经解: string; " +
//                  "columnName: uk:  崇源: string; " +
//                  "columnName: uk:  愚按: string; " +
//                  "columnName: uk:  搭配: string; " +
//                  "columnName: uk:  常见药: string; ";
//这个函数用于确定Deta PLSQL的Root like: 'c: /dsdsd' etc String
    PLSQL= "";
    public String startAtRootDir(String rootAddress) {
```

```
        PLSQL= "setRoot: "+ rootAddress+ "; ";
        return PLSQL;
    }
    public String withBaseName(String baseName) { PLSQL+=
        "baseName: "+ baseName+ "; "; return PLSQL;
    }
    public String withTableCreate(String tableName) { PLSQL+=
        "tableName: "+ tableName+ ": create; "; return PLSQL;
    }
    public String withCulumnName(String culumnName, String keyType, String dataFormat) { PLSQL+=
        "culumnName: "+ keyType+ ": "+ culumnName+ ": "+ dataFormat+ "; "; return PLSQL;
    }
}
```

Dictionary PLSQL Standard

```
package OSV.VCQ.standard;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import OSI.AOP.MEC.SIQ.plorm.PLORM_E;
import OSI.AOP.MEC.SIQ.plorm.PLORM_C;
//OSV PLSQL 研发, 思想, 设计: 罗瑶光 20210506
public class DictionaryPLSQLStandardDB{
    public List<String> txtToList() throws IOException{
        List<String> dic_list= new ArrayList<>();
        return dic_list;
    }

    public static Map<String, Object> bootORMReadDBInCommonWay(String tabKey) throws IOException{
        Map<String, Object> map= null;
        try {
            PLORM_C orm= new PLORM_E();
            map= orm.startAtRootDir("C:/DetaDB1").withBaseName("ZYY")
                    .withTableSelect(tabKey).withCondition("or")
                    .let("ID").lessThanAndEqualTo("3000")
                    .checkAndFixPlsqlGrammarErrors()//准备完善plsql orm语言 的语法检查函数 和修复函数。
                    .checkAndFixSystemEnvironmentErrors()//准备完善plsql orm语言 的系统环境检查函数和修复函
数。
                    .finalE(true).returnAsMap();
            //map= org.plsql.db.plsql.imp.E_PLSQLImp.E_PLORM(orm, true);
        }catch(Exception e1) {
            //准备写回滚
            e1.printStackTrace();
        }
        return map;
    }
```

```
	public static Map<String, Object> bootORMReadDBByRangeRowID(String rootPath, String baseName
			, boolean unTest, String tabKey, String RangeRowIDCount) throws IOException{
		Map<String, Object> map= null;
		try {
			PLORM_C orm= new PLORM_E();
			map= orm.startAtRootDir(rootPath).withBaseName(baseName)
					.withTableSelect(tabKey).withCondition("or")
					.let("ID").lessThanAndEqualTo(RangeRowIDCount)
					.checkAndFixPlsqlGrammarErrors()//准备完善plsql orm语言 的语法检查函数 和修复函数。
					.checkAndFixSystemEnvironmentErrors()//准备完善plsql orm语言 的系统环境检查函数和修复函
数。
					.finalE(unTest).returnAsMap();
			//map= org.plsql.db.plsql.imp.E_PLSQLImp.E_PLORM(orm, true);
		}catch(Exception e1) {
			//准备写回滚
			e1.printStackTrace();
		}
		return map;
	}
	//下面这些例子, 本人只是给大家一些更多的参考而已.
	public static Map<String, Object> bootORMReadDBByLessThanAndEqualTo(String rootPath, String baseName
			, String conditionSubject, String conditionObject, boolean unTest, String tabKey) throws IOException{
		Map<String, Object> map= null;
		try {
			PLORM_C orm= new PLORM_E();
			map= orm.startAtRootDir(rootPath).withBaseName(baseName)
					.withTableSelect(tabKey).withCondition("or")
					.let(conditionSubject).lessThanAndEqualTo(conditionObject)
					.checkAndFixPlsqlGrammarErrors()//准备完善plsql orm语言 的语法检查函数 和修复函数。
					.checkAndFixSystemEnvironmentErrors()//准备完善plsql orm语言 的系统环境检查函数和修复函
数。
					.finalE(unTest).returnAsMap();
			//map= org.plsql.db.plsql.imp.E_PLSQLImp.E_PLORM(orm, true);
		}catch(Exception e1) {
			//准备写回滚
			e1.printStackTrace();
		}
		return map;
	}

	public static Map<String, Object> bootPLSQLReadDBInCommonWay(String tabKey) throws IOException{
		Map<String, Object> map= null;
		try {
			String plsql= "setRoot:C:/DetaDB1;" +
					"baseName:ZYY;" +
					"tableName:"+ tabKey +":select;" +
					"condition:or:ID|<=|3000;";
			map= ME.SM.OP.SM.AOP.MEC.SIQ.E.E_PLSQL_E.E_PLSQL(plsql, true);
		}catch(Exception e1) {
			//准备写回滚
			e1.printStackTrace();
		}
		return map;
	}
}
```

```
package ME.SM.OP.SM.AOP.MEC.SIQ.E;
import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;
```

```
import OSI.AOP.MEC.SIQ.plorm.PLORM_C;
public class E_PLSQL_E {
 public static Map<String, Object> E_PLSQL(String plsql, boolean mod) throws Exception{
		//working for here
		Map<String, Object> output = new ConcurrentHashMap<>();
		//1make container
		output.put("start", "0");
		output.put("countJoins", "0");
		//2make line
		String[] commands = plsql.replace(" ", "").replace("\n", "").split(";");
		String[] acknowledge = null;
		for(String command:commands) {
			acknowledge = command.split(":");
			if(acknowledge[0].equals("setRoot")) {
				PLSQLCommand_E.P_SetRoot(acknowledge, output);
			}
			if(acknowledge[0].equals("baseName")) {
				PLSQLCommand_E.P_BaseName(acknowledge, output);
			}
			if(acknowledge[0].equals("tableName")) {
				PLSQLCommand_E.P_TableName(acknowledge, output);
			}
			if(acknowledge[0].equals("culumnName")) {
				PLSQLCommand_E.P_ListNeedStart(acknowledge, output);
			}
			if(acknowledge[0].equals("changeCulumnName")) {
				PLSQLCommand_E.P_ListNeedStart(acknowledge, output);
			}
			if(acknowledge[0].equals("culumnValue")) {
				PLSQLCommand_E.P_ListNeedStart(acknowledge, output);
			}
			if(acknowledge[0].equals("join")) {
				PLSQLCommand_E.P_Join(acknowledge, output);
			}
			if(acknowledge[0].equals("condition")) {
				PLSQLCommand_E.P_ListNeedStart(acknowledge, output);
			}
			if(acknowledge[0].equals("relation")) {
				PLSQLCommand_E.P_ListNeedStart(acknowledge, output);
			}
			if(acknowledge[0].equals("aggregation")) {
				PLSQLCommand_E.P_ListNeedStart(acknowledge, output);
			}
			if(acknowledge[0].equals("getCulumns")) {
				PLSQLCommand_E.P_ListNeedStart(acknowledge, output);
			}
			output.put("newCommand", acknowledge[0]);
			PLSQLCommand_E.P_E(acknowledge, output, mod);
			output.put("lastCommand", output.get("newCommand"));
		}
		if(null!= acknowledge) {
			if(output.get("start").toString().equals("1")) {
				PLSQLCommand_E.P_E(acknowledge, output, mod);
			}
		}
		System.out.println("1");
		PLSQLCommand_E.P_Check(output.get("newCommand").toString(), output, mod);
		return output;
 }

 public static Map<String, Object> E_PLORM(PLORM_C orm, boolean b) throws Exception {
		return E_PLSQL(orm.getPLSQL(), true);
```

```
}
}
```

```
package ME.SM.OP.SM.AOP.MEC.SIQ.E;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import MS.OP.SM.AOP.MEC.SIQ.cache.DetaDBBufferCache_M;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Cell;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Row;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Table;
@SuppressWarnings({ "unused", "unchecked"})
public class P_ConditionPLSQL_XCDX_Cache extends P_ConditionPLSQL_XCDX {
 public static void P_Cache(String[] sets, List<Map<String, Object>> output
                , String tableName, String baseName, Map<String, Object> object) {
        Table table= DetaDBBufferCache_M.db.getBase(baseName).getTable(tableName);
        Iterator<String> iterator= table.getRows().keySet().iterator();
        int rowindex=0;
        while(iterator.hasNext()) {
                int count= rowindex++;
                String rowIndex= iterator.next();
                Row row= table.getRow(rowIndex);
                Cell cell=new Cell();
                cell.I_CellValue(rowIndex.replace("row", ""));
                row.putCell("Index", cell);
                if(sets[1].equalsIgnoreCase("<")||sets[1].equalsIgnoreCase("-lt")) {
                        double rowCellFromBigDecimal= new BigDecimal(row.getCell(sets[0])
                                        .getCellValue().toString()).doubleValue();
                        if(rowCellFromBigDecimal< new BigDecimal(sets[2]).doubleValue()
                                        && row.containsCell("is_delete_0")) {
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(row));
                                        Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>)
object.get("recordRows");
                                        recordRows.put(count, true);
                                }
                        }
                }
                if(sets[1].equalsIgnoreCase("<=")||sets[1].equalsIgnoreCase("=<")
                                ||sets[1].equalsIgnoreCase("-lte")) {

                        String set= sets[0];
                        Cell setCell= row.getCell(set);
                        String cellString= setCell.getCellValue().toString();
                        cellString=cellString.isEmpty()? "0": cellString;
                        double rowCellFromBigDecimal = new BigDecimal(cellString).doubleValue();
                        if(rowCellFromBigDecimal<=  new BigDecimal(sets[2]).doubleValue()
                                        && row.containsCell("is_delete_0")) {
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(row));
                                        Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>)
object.get("recordRows");
```

```
                                        recordRows.put(count, true);
                                }
                        }
                }
                if(sets[1].equalsIgnoreCase("==")||sets[1].equalsIgnoreCase("=")
                                ||sets[1].equalsIgnoreCase("===")) {
                        double rowCellFromBigDecimal = new BigDecimal(row.getCell(sets[0])
                                        .getCellValue().toString()).doubleValue();
                        if(rowCellFromBigDecimal ==  new BigDecimal(sets[2]).doubleValue()
                                        && row.containsCell("is_delete_0")) {
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(row));
                                        Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>)
object.get("recordRows");
                                        recordRows.put(count, true);
                                }
                        }
                }
                if(sets[1].equalsIgnoreCase(">=")||sets[1].equalsIgnoreCase("=>")
                                ||sets[1].equalsIgnoreCase("-gte")) {
                        double rowCellFromBigDecimal = new BigDecimal(row.getCell(sets[0])
                                        .getCellValue().toString()).doubleValue();
                        if(rowCellFromBigDecimal >= new BigDecimal(sets[2]).doubleValue()
                                        && row.containsCell("is_delete_0")) {
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(row));
                                        Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>)
object.get("recordRows");
                                        recordRows.put(count, true);
                                }
                        }
                }
                if(sets[1].equalsIgnoreCase(">")||sets[1].equalsIgnoreCase("-gt")) {
                        double rowCellFromBigDecimal = new BigDecimal(row.getCell(sets[0])
                                        .getCellValue().toString()).doubleValue();
                        if(rowCellFromBigDecimal > new BigDecimal(sets[2]).doubleValue()
                                        && row.containsCell("is_delete_0")) {
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(row));
                                        Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>)
object.get("recordRows");
                                        recordRows.put(count, true);
                                }
                        }
                }
                if(sets[1].equalsIgnoreCase("!=")||sets[1].equalsIgnoreCase("=!")) {
                        double rowCellFromBigDecimal = new BigDecimal(row.getCell(sets[0])
                                        .getCellValue().toString()).doubleValue();
                        if(rowCellFromBigDecimal != new BigDecimal(sets[2]).doubleValue()
                                        && row.containsCell("is_delete_0")) {
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(row));
                                        Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>)
object.get("recordRows");
                                        recordRows.put(count, true);
                                }
                        }
                }
                if(sets[1].equalsIgnoreCase("equal") && row.containsCell("is_delete_0")) {
                        String rowCellFromString = row.getCell(sets[0]).getCellValue().toString();
                        if(rowCellFromString.equalsIgnoreCase(sets[2])) {
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
```

```
                                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(row));
                                        Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>)
object.get("recordRows");
                                        recordRows.put(count, true);
                                }
                        }
                }
                if(sets[1].equalsIgnoreCase("!equal")) {
                        String rowCellFromString = row.getCell(sets[0]).getCellValue().toString();
                        if(!rowCellFromString.equalsIgnoreCase(sets[2]) && row.containsCell("is_delete_0")) {
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(row));
                                        Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>)
object.get("recordRows");
                                        recordRows.put(count, true);
                                }
                        }
                }

                if(sets[1].equalsIgnoreCase("in")) {
                        String rowCellFromString = row.getCell(sets[0]).getCellValue().toString();
                        String set = "," + sets[2] + ",";
                        if(set.contains("," + rowCellFromString + ",") && row.containsCell("is_delete_0")){
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(row));
                                        Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>)
object.get("recordRows");
                                        recordRows.put(count, true);
                                }
                        }
                }
                if(sets[1].equalsIgnoreCase("!in")) {
                        String rowCellFromString = row.getCell(sets[0]).getCellValue().toString();
                        String set = "," + sets[2] + ",";
                        if(!set.contains("," + rowCellFromString + ",") && row.containsCell("is_delete_0")){
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(row));
                                        Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>)
object.get("recordRows");
                                        recordRows.put(count, true);
                                }
                        }
                }
        }
 }
}
```

```
package ME.SM.OP.SM.AOP.MEC.SIQ.E;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.List;
import java.util.Map;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Cell;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Row;
public class P_ConditionPLSQL_XCDX_Kernel extends P_ConditionPLSQL_XCDX {
//比较是否有数据取出列表到输出 检验中 罗瑶光 20210405
        //这个走硬盘查询函数来标识下, 在我设计了数据缓存查询启动函数 后就没用过了, 时间点大概在2019年1
月后, 我先调通下, 之后朔源.
        //准备验算下20210406 罗瑶光
```

```
	public static void P_kernel(String temp, File readDBTableRowIndexCulumnFile, File
readDBTableRowIndexFile
			, BufferedReader reader, String DBTableRowIndexPath, List<Map<String, Object>> output,
Row bufferRow
			, Map<String, Object> rowMap) throws IOException {
		String[] culumnList = readDBTableRowIndexFile.list();
		NextFile:
			for(String culumn: culumnList) {
				if(culumn.contains("is_delete")) {
					continue NextFile;
				}
				String DBTableCulumnIndexPath = readDBTableRowIndexFile + "/" + culumn;
				File readDBTableCulumnIndexPathFile = new File(DBTableCulumnIndexPath);

				Cell cell= new Cell();
				if (readDBTableCulumnIndexPathFile.isDirectory()) {
					//似乎被动了手脚, 20210405 罗瑶光重新检查
					reader = new BufferedReader(new
FileReader(readDBTableCulumnIndexPathFile + "/" + "value.lyg"));
					temp = "";
					String tempString;
					while ((tempString = reader.readLine()) != null) {
						temp += tempString;
					}
					reader.close();
					rowMap.put(culumn, temp);
					cell.I_CellValue(temp);
					bufferRow.putCell(culumn, cell);
				}else {
					rowMap.put(culumn, null);
					cell.I_CellValue(null);
					bufferRow.putCell(culumn, cell);
				}
			}
		output.add(rowMap);
	}
}
```

```
package ME.SM.OP.SM.AOP.MEC.SIQ.E;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import MS.OP.SM.AOP.MEC.SIQ.cache.DetaDBBufferCache_M;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Cell;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Row;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Table;
@SuppressWarnings({ "unused", "unchecked"})
public class P_ConditionPLSQL_XCDX_Map extends P_ConditionPLSQL_XCDX {
//以后优化成统一对象输出，不需要再转换。2019-1-15 tin
 public static Map<String, Object> rowToRowMap(Row row) {
	Map<String, Object> culumnMaps= new HashMap<>();
	Map<String, Object> rowMap= new HashMap<>();
```

```
        Iterator<String> iterator= row.getCells().keySet().iterator();
        while(iterator.hasNext()) {
                String cellName = iterator.next();
                if(!cellName.contains("is_delete")) {
                        Cell cell = row.getCell(cellName);
                        Map<String, Object> culumnMap = new HashMap<>();
                        culumnMap.put("culumnName", cellName);
                        culumnMap.put("culumnValue", cell.getCellValue().toString());
                        culumnMaps.put(cellName, culumnMap);
                }
        }
        rowMap.put("rowValue", culumnMaps);
        return rowMap;
}

public static void P_Map(String[] sets, List<Map<String, Object>> output, String dBTablePath) {
        List<Map<String, Object>> outputTemp = new ArrayList<>();
        Iterator<Map<String, Object>> iterator = output.iterator();
        int rowid = 0;
        while(iterator.hasNext()) {
                Map<String, Object> row = iterator.next();
                Map<String, Object> rowMap = new HashMap<>();
                if(sets[1].equalsIgnoreCase("<")||sets[1].equalsIgnoreCase("-lt")) {
                        String rowCellFromString = ((Map<String, Object>)(((Map<String,
Object>)(row.get("rowValue")))
                                        .get(sets[0]))).get("culumnValue").toString();
                        if(new BigDecimal(rowCellFromString).doubleValue() < new
BigDecimal(sets[2]).doubleValue()) {
                                outputTemp.add(row);
                        }
                }
                if(sets[1].equalsIgnoreCase("<=")||sets[1].equalsIgnoreCase("=<")
                                ||sets[1].equalsIgnoreCase("-lte")) {
                        String rowCellFromString = ((Map<String, Object>)(((Map<String,
Object>)(row.get("rowValue")))
                                        .get(sets[0]))).get("culumnValue").toString();
                        if(new BigDecimal(rowCellFromString).doubleValue() <=  new
BigDecimal(sets[2]).doubleValue()) {
                                outputTemp.add(row);
                        }
                }
                if(sets[1].equalsIgnoreCase("==")||sets[1].equalsIgnoreCase("=")||sets[1].equalsIgnoreCase("===")) {
                        String rowCellFromString = ((Map<String, Object>)(((Map<String,
Object>)(row.get("rowValue")))
                                        .get(sets[0]))).get("culumnValue").toString();
                        if(new BigDecimal(rowCellFromString).doubleValue() ==  new
BigDecimal(sets[2]).doubleValue()) {
                                outputTemp.add(row);
                        }
                }
                if(sets[1].equalsIgnoreCase(">=")||sets[1].equalsIgnoreCase("=>")
                                ||sets[1].equalsIgnoreCase("-gte")) {
                        String rowCellFromString = ((Map<String, Object>)(((Map<String,
Object>)(row.get("rowValue")))
                                        .get(sets[0]))).get("culumnValue").toString();
                        if(new BigDecimal(rowCellFromString).doubleValue() >= new
BigDecimal(sets[2]).doubleValue()) {
                                outputTemp.add(row);
                        }
                }
                if(sets[1].equalsIgnoreCase(">")||sets[1].equalsIgnoreCase("-gt")) {
                        String rowCellFromString = ((Map<String, Object>)(((Map<String,
```

```
Object>)(row.get("rowValue")))
                                        .get(sets[0])).get("culumnValue").toString();
                        if(new BigDecimal(rowCellFromString).doubleValue() > new
BigDecimal(sets[2]).doubleValue()) {
                                outputTemp.add(row);
                        }
                }
                if(sets[1].equalsIgnoreCase("!=")||sets[1].equalsIgnoreCase("=!")) {
                        String rowCellFromString = ((Map<String, Object>)(((Map<String,
Object>)(row.get("rowValue")))
                                        .get(sets[0])).get("culumnValue").toString();
                        if(new BigDecimal(rowCellFromString).doubleValue() != new
BigDecimal(sets[2]).doubleValue()) {
                                outputTemp.add(row);
                        }
                }

                if(sets[1].equalsIgnoreCase("equal")) {
                        String rowCellFromString = ((Map<String, Object>)(((Map<String,
Object>)(row.get("rowValue")))
                                        .get(sets[0])).get("culumnValue").toString();
                        if(rowCellFromString.equalsIgnoreCase(sets[2])) {
                                outputTemp.add(row);
                        }
                }

                if(sets[1].equalsIgnoreCase("!equal")) {
                        String rowCellFromString = ((Map<String, Object>)(((Map<String,
Object>)(row.get("rowValue")))
                                        .get(sets[0])).get("culumnValue").toString();
                        if(!rowCellFromString.equalsIgnoreCase(sets[2])) {
                                outputTemp.add(row);
                        }
                }

                if(sets[1].equalsIgnoreCase("in")) {
                        String rowCellFromString = ((Map<String, Object>)(((Map<String,
Object>)(row.get("rowValue")))
                                        .get(sets[0])).get("culumnValue").toString();
                        String set = "," + sets[2] + ",";
                        if(set.contains("," + rowCellFromString + ",")){
                                outputTemp.add(row);
                        }
                }

                if(sets[1].equalsIgnoreCase("!in")) {
                        String rowCellFromString = ((Map<String, Object>)(((Map<String,
Object>)(row.get("rowValue")))
                                        .get(sets[0])).get("culumnValue").toString();
                        String set = "," + sets[2] + ",";
                        if(!set.contains("," + rowCellFromString + ",")){
                                outputTemp.add(row);
                        }
                }
        }
        output.clear();
        output.addAll(outputTemp);
 }
}
```

```
package ME.SM.OP.SM.AOP.MEC.SIQ.E;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import MS.OP.SM.AOP.MEC.SIQ.cache.DetaDBBufferCache_M;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Cell;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Row;
import OP.SM.AOP.MEC.SIQ.SM.reflection.Table;
@SuppressWarnings({ "unused", "unchecked"})
public class P_ConditionPLSQL_XCDX_Table extends P_ConditionPLSQL_XCDX {
//plsql引擎函数获取表开始检查 罗瑶光 20210405  //奇怪了 这是一个没有读 缓存的plsql引擎,我准备对比下history
//object 指令堆栈
//output 数据行
public static void P_Table(String[] sets, List<Map<String, Object>> output
                        , String DBTablePath, Map<String, Object> object) throws IOException {
        String DBTableRowsPath= DBTablePath + "/rows";
        File fileDBTableRowsPath= new File(DBTableRowsPath);
        if (fileDBTableRowsPath.isDirectory()) {
                String[] rowList= fileDBTableRowsPath.list();
                int count= 0;
                NextRow:
                for(String row: rowList) {
                        count++;
                        Map<String, Object> rowMap= new HashMap<>();
                        String DBTableRowIndexPath= DBTablePath + "/rows/" + row;
                        File readDBTableRowIndexFile= new File(DBTableRowIndexPath);
                        if (readDBTableRowIndexFile.isDirectory()) {
                                String isDelete= DBTableRowIndexPath + "/is_delete_1" ;
                                File isDeleteFile= new File(isDelete);
                                if(isDeleteFile.exists()) {
                                        continue NextRow;
                                }
                                String DBTableRowIndexCulumnPath= DBTableRowIndexPath + "/" + sets[0];
                                File readDBTableRowIndexCulumnFile= new File(DBTableRowIndexCulumnPath);
                                if(readDBTableRowIndexCulumnFile.isDirectory()) {
                                        BufferedReader reader= new BufferedReader
                                                                        (new FileReader(readDBTableRowIndexCulumnFile +
"/" + "value.lyg"));
                                        String temp= "";
                                        String tempString= "";
                                        while ((tempString= reader.readLine())!= null) {
                                                temp+= tempString;
                                        }
                                        reader.close();
                                        if(temp.isEmpty()) {//增加一行id为空检查, 大家记得给 数据库的id加点值,我lyg的都是
空文件.
                                                continue NextRow;
                                        }
                                        if(sets[1].equalsIgnoreCase("<")|| sets[1].equalsIgnoreCase("-lt")) {
        if(new BigDecimal(temp.toString()).doubleValue() < new BigDecimal(sets[2].toString()).doubleValue()) {
                                        if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
```

```
                                        Row bufferRow= new Row();
                                                bufferRow.I_Cells(new ConcurrentHashMap<String, Cell>());
                                                        P_ConditionPLSQL_XCDX_Kernel.P_kernel(row,
readDBTableRowIndexCulumnFile, readDBTableRowIndexFile, reader// 似乎被猫腻哥动了手脚, 我会将手里硬盘数据2年的数据等会全部验算
, row, output, bufferRow, rowMap);output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(bufferRow));
        Map<Integer, Boolean> recordRows = (Map<Integer, Boolean>) object.get("recordRows"); recordRows.put(count, true);

                                        }
                                }
                        }

        if(sets[1].equalsIgnoreCase("<=")||sets[1].equalsIgnoreCase("=<")||sets[1].equalsIgnoreCase("-lte")) {
                if(new BigDecimal(temp.toString()).doubleValue() <= new BigDecimal(sets[2].toString()).doubleValue()) {
                        if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                Row bufferRow= new Row();
                                bufferRow.I_Cells(new ConcurrentHashMap<String, Cell>());
                                P_ConditionPLSQL_XCDX_Kernel.P_kernel(row, readDBTableRowIndexCulumnFile
                , readDBTableRowIndexFile, reader// 似乎被猫腻哥动了手脚, 我会将手里硬盘数据2年的数据等会全部验算
                                , row, output, bufferRow, rowMap);
                                output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(bufferRow));
                                Map<Integer, Boolean> recordRows = (Map<Integer, Boolean>) object.get("recordRows");
                                recordRows.put(count, true);
                                }
                        }
                }
        if(sets[1].equalsIgnoreCase("==")||sets[1].equalsIgnoreCase("=")||sets[1].equalsIgnoreCase("===")) {
                if(new BigDecimal(temp.toString()).doubleValue() == new BigDecimal(sets[2].toString()).doubleValue()) {
                        if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                Row bufferRow= new Row();
                                bufferRow.I_Cells(new ConcurrentHashMap<String, Cell>());
                                P_ConditionPLSQL_XCDX_Kernel.P_kernel(row, readDBTableRowIndexCulumnFile
                , readDBTableRowIndexFile, reader// 似乎被猫腻哥动了手脚, 我会将手里硬盘数据2年的数据等会全部验算
                        , row, output, bufferRow, rowMap);
                                output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(bufferRow));
                                        Map<Integer, Boolean> recordRows = (Map<Integer, Boolean>) object.get("recordRows");
                                        recordRows.put(count, true);
                                }
                        }
                }
                if(sets[1].equalsIgnoreCase(">=")||sets[1].equalsIgnoreCase("=>") ||sets[1].equalsIgnoreCase("-gte")) {
                        if(new BigDecimal(temp.toString()).doubleValue() >= new BigDecimal(sets[2].toString()).doubleValue()) {
                                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                        Row bufferRow= new Row();
                                        bufferRow.I_Cells(new ConcurrentHashMap<String, Cell>());
                                        P_ConditionPLSQL_XCDX_Kernel.P_kernel(row, readDBTableRowIndexCulumnFile
                , readDBTableRowIndexFile, reader// 似乎被猫腻哥动了手脚, 我会将手里硬盘数据2年的数据等会全部验算
                        , row, output, bufferRow, rowMap);
                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(bufferRow));
                        Map<Integer, Boolean> recordRows = (Map<Integer, Boolean>) object.get("recordRows");
                        recordRows.put(count, true);
                        }
                }
        }
        if(sets[1].equalsIgnoreCase(">")||sets[1].equalsIgnoreCase("-gt")) {
                if(new BigDecimal(temp.toString()).doubleValue() > new BigDecimal(sets[2].toString()).doubleValue()) {
                        if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                Row bufferRow= new Row();
                                bufferRow.I_Cells(new ConcurrentHashMap<String, Cell>());
                                P_ConditionPLSQL_XCDX_Kernel.P_kernel(row, readDBTableRowIndexCulumnFile
                , readDBTableRowIndexFile, reader// 似乎被猫腻哥动了手脚, 我会将手里硬盘数据2年的数据等会全部验算
```

```
                , row, output, bufferRow, rowMap);
            output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(bufferRow));
            Map<Integer, Boolean> recordRows = (Map<Integer, Boolean>) object.get("recordRows");
                recordRows.put(count, true);
                }
            }
    }
    if(sets[1].equalsIgnoreCase("!=")||sets[1].equalsIgnoreCase("=!")) {
            if(new BigDecimal(temp.toString()).doubleValue()!= new BigDecimal(sets[2].toString()).doubleValue()) {
                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                        Row bufferRow= new Row();
                        bufferRow.I_Cells(new ConcurrentHashMap<String, Cell>());
                        P_ConditionPLSQL_XCDX_Kernel.P_kernel(row, readDBTableRowIndexCulumnFile
            , readDBTableRowIndexFile, reader// 似乎被猫腻哥动了手脚, 我会将手里硬盘数据2年的数据等会全部验算
                , row, output, bufferRow, rowMap);
            output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(bufferRow));
            Map<Integer, Boolean> recordRows = (Map<Integer, Boolean>) object.get("recordRows");
            recordRows.put(count, true);
                }
            }
    }
    if(sets[1].equalsIgnoreCase("equal")) {
            String rowCellFromString = temp.toString();
            if(rowCellFromString.equalsIgnoreCase(sets[2].toString())) {
                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                        Row bufferRow= new Row();
                        bufferRow.I_Cells(new ConcurrentHashMap<String, Cell>());
                        P_ConditionPLSQL_XCDX_Kernel.P_kernel(row, readDBTableRowIndexCulumnFile
            , readDBTableRowIndexFile, reader// 似乎被猫腻哥动了手脚, 我会将手里硬盘数据2年的数据等会全部验算
                , row, output, bufferRow, rowMap);
            output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(bufferRow));
            Map<Integer, Boolean> recordRows = (Map<Integer, Boolean>) object.get("recordRows");
                recordRows.put(count, true);
                }
            }
    }
    if(sets[1].equalsIgnoreCase("!equal")) {
            String rowCellFromString = temp.toString();
            if(!rowCellFromString.equalsIgnoreCase(sets[2].toString())) {
                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                        Row bufferRow= new Row();
                        bufferRow.I_Cells(new ConcurrentHashMap<String, Cell>());
                        P_ConditionPLSQL_XCDX_Kernel.P_kernel(row, readDBTableRowIndexCulumnFile
            , readDBTableRowIndexFile, reader// 似乎被猫腻哥动了手脚, 我会将手里硬盘数据2年的数据等会全部验算
                , row, output, bufferRow, rowMap);
                output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(bufferRow));
            Map<Integer, Boolean> recordRows = (Map<Integer, Boolean>) object.get("recordRows");
            recordRows.put(count, true);
                }
            }
    }
    if(sets[1].equalsIgnoreCase("in")) {
            String rowCellFromString = temp.toString();
            String set = "," + sets[2] + ",";
            if(set.contains("," + rowCellFromString + ",")) {
                if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                        Row bufferRow= new Row();
                        bufferRow.I_Cells(new ConcurrentHashMap<String, Cell>());
                        P_ConditionPLSQL_XCDX_Kernel.P_kernel(row, readDBTableRowIndexCulumnFile
            , readDBTableRowIndexFile, reader// 似乎被猫腻哥动了手脚, 我会将手里硬盘数据2年的数据等会全部验算
                , row, output, bufferRow, rowMap);
                output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(bufferRow));
```

```
                        Map<Integer, Boolean> recordRows = (Map<Integer, Boolean>) object.get("recordRows");
                        recordRows.put(count, true);
                        }
                }
        }
        if(sets[1].equalsIgnoreCase("!in")) {
                String rowCellFromString = temp.toString();
                String set = "," + sets[2] + ",";
                if(!set.contains("," + rowCellFromString + ",")) {
                        if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(count)) {
                                Row bufferRow= new Row();
                                bufferRow.I_Cells(new ConcurrentHashMap<String, Cell>());
                                P_ConditionPLSQL_XCDX_Kernel.P_kernel(row, readDBTableRowIndexCulumnFile
                , readDBTableRowIndexFile, reader// 似乎被猫腻哥动了手脚, 我会将手里硬盘数据2年的数据等会全部验算
                                , row, output, bufferRow, rowMap);
                        output.add(P_ConditionPLSQL_XCDX_Map.rowToRowMap(bufferRow));
                        Map<Integer, Boolean> recordRows = (Map<Integer, Boolean>) object.get("recordRows");
                                recordRows.put(count, true);
                                        }
                                }
                        }
                }
        }
}
```

Deta DNA Index & PLSQL ORM 增删改查 Demo

```
package OSV.ESD.standard;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
//import OSI.AOP.MEC.SIQ.plorm.PLORMImpl;
//import OSI.AOP.MEC.SIQ.plorm.PLORMInterf;
//import OSI.OPE.SI.SD.SU.SQ.ASU.OSU.PSU.MSU.AVQ.ASQ.ASU.MPE.procedure.pde.FullDNATokenPDI;
//OSV PLSQL 研发，思想，设计：罗瑶光 20210511
//表删除操作的例子.
public class D_CellStandard{
    public List<String> txtToList() throws IOException{
        List<String> dic_list= new ArrayList<>();
        return dic_list;
    }
    //    delete samples
    //    tableName:test:delete;
    //    condition:or:testCulumn1|<|20:testCulumn2|==|fire;
    //    condition:and:testCulumn1|>|100:testCulumn2|==|fire;
    public static Map<String, Object> DeleteCellORM(String rootPath, String baseName
            , String tabKey
            , Map<String, String> rowCells, Boolean initonEncrypt) throws IOException{
        //   PLORMInterf orm= new PLORMImpl();
        //    orm= orm.startAtRootDir(rootPath)
        //                .withBaseName(baseName)
        //                .withTableInsert(tabKey);
        //   condition 不好规范，稍后．建议写针对性 delete 语句
        //    condition:or:testCulumn1|<|20:testCulumn2|==|fire;
        //    condition:and:testCulumn1|>|100:testCulumn2|==|fire;
        return null;
    }
    public static Map<String, Object> DeleteCellPLSQL(String rootPath, String baseName
            , String tabKey
            , Map<String, String> rowCells, Boolean initonEncrypt) throws IOException{
        Map<String, Object> map= null;
        try {
            //    String plsql= "setRoot:"+ rootPath+ ";";
            //    plsql+= "baseName:"+ baseName+ ";";
            //    plsql+= "tableName:"+ tabKey+ ":delete;";
            //   condition 不好规范，稍后．建议写针对性 delete 语句
            //    condition:or:testCulumn1|<|20:testCulumn2|==|fire;
            //    condition:and:testCulumn1|>|100:testCulumn2|==|fire;
            //    map= OSI.OPE.ME.SM.OP.SM.AOP.MEC.SIQ.imp.E_PLSQLImp.E_PLSQL(plsql, true);
        }catch(Exception e1) {
            //准备写回滚
```

```
            e1.printStackTrace();
        }
        return map;
    }
}
```

```
package OSV.ESD.standard;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import OSI.AOP.MEC.SIQ.plorm.PLORM_E;
import OSI.AOP.MEC.SIQ.plorm.PLORM_C;
//import OSI.OPE.SI.SD.SU.SQ.ASU.OSU.PSU.MSU.AVQ.ASQ.ASU.MPE.procedure.pde.FullDNATokenPDI;
//OSV PLSQL 研发，思想，设计：罗瑶光 20210511
//表剔除操作的例子.
public class D_Standard{
    public List<String> txtToList() throws IOException{
        List<String> dic_list= new ArrayList<>();
        return dic_list;
    }
    //    drop samples
    //    tableName:test:drop;
    public static Map<String, Object> DropCellORM(String rootPath, String baseName
            , String tabKey) throws Exception{
        PLORM_C orm= new PLORM_E();
        orm= orm.startAtRootDir(rootPath)
                .withBaseName(baseName)
                .withTableDrop(tabKey)
                .checkAndFixPlsqlGrammarErrors()
                //准备完善 plsql orm 语言 的语法检查函数 和修复函数。
                .checkAndFixSystemEnvironmentErrors()
                //准备完善 plsql orm 语言 的系统环境检查函数和修复函数。
                .finalE(true);
        return null;
    }
    public static Map<String, Object> DropCellPLSQL(String rootPath, String baseName
            , String tabKey) throws IOException{
        Map<String, Object> map= null;
        try {
            String plsql= "setRoot:"+ rootPath+ ";";
            plsql+= "baseName:"+ baseName+ ";";
            plsql+= "tableName:"+ tabKey+ ":drop;";
            map= ME.SM.OP.SM.AOP.MEC.SIQ.E.E_PLSQL_E.E_PLSQL(plsql, true);
        }catch(Exception e1) {
            //准备写回滚
            e1.printStackTrace();
```

```
        }
        return map;
    }
}
```

```
package OSV.ESI.standard;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import OSI.AOP.MEC.SIQ.plorm.PLORM_E;
import OSI.AOP.MEC.SIQ.plorm.PLORM_C;
import OSI.OPE.SI.SD.SU.SQ.ASU.OSU.PSU.MSU.AVQ.ASQ.ASU.MPE.procedure.pde.FullDNATokenPDI;
//OSV PLSQL 研发，思想，设计：罗瑶光 20210510
//插入一行完整的数据.
//我没有测试，现在养疗经还用不到，优先级稍后.
//稍后会考虑不完整的插入情况 赋值 empty 或 null.
public class IU_CellStandard{
    public List<String> txtToList() throws IOException{
        List<String> dic_list= new ArrayList<>();
        return dic_list;
    }
    public static Map<String, Object> InsertCellORM(String rootPath, String baseName
            , String tabKey
            , Map<String, String> rowCells, Boolean initonEncrypt) throws IOException{
        Iterator<String> iterators= rowCells.keySet().iterator();
        Map<String, Object> map= null;
        try {
            PLORM_C orm= new PLORM_E();
            orm= orm.startAtRootDir(rootPath)
                    .withBaseName(baseName)
                    .withTableInsert(tabKey);
            while(iterators.hasNext()) {
                String string= iterators.next();
                if(initonEncrypt) {
                    orm= orm.withCulumnValue(string
                            , new FullDNATokenPDI().initonSect
                            (rowCells.get(string).replace(":", "@Tin@")));
                }else {
                    orm= orm.withCulumnValue(string, rowCells.get(string).replace(":"
                            , "@Tin@"));
                    //我稍后会思考: 符号怎么进行实体化,省的要变通配符代替.
                }
            }
            orm.checkAndFixPlsqlGrammarErrors()
            //准备完善 plsql orm 语言 的语法检查函数 和修复函数。
```

```
                .checkAndFixSystemEnvironmentErrors()
                //准备完善 plsql orm 语言 的系统环境检查函数和修复函数。
                .finalE(true);
                //map= org.plsql.db.plsql.imp.E_PLSQLImp.E_PLORM(orm, true);
        }catch(Exception e1) {
                //准备写回滚
                e1.printStackTrace();
        }
        return map;
    }
    public static Map<String, Object> InsertCellPLSQL(String rootPath, String baseName
                , String tabKey
                , Map<String, String> rowCells, Boolean initonEncrypt) throws IOException{
        Iterator<String> iterators= rowCells.keySet().iterator();
        Map<String, Object> map= null;
        try {
                String plsql= "setRoot:"+ rootPath+ ";";
                plsql+= "baseName:"+ baseName+ ";";
                plsql+= "tableName:"+ tabKey+ ":insert;";
                while(iterators.hasNext()) {
                        String string= iterators.next();
                        if(initonEncrypt) {
                                plsql+= "culumnValue:"+ string+ ":"
                                                + new FullDNATokenPDI().initonSect
                                                (rowCells.get(string).replace(":", "@Tin@"))+ ";";
                        }else {
                                plsql+= "culumnValue:"+ string+ ":"
                        + rowCells.get(string).replace(":", "@Tin@")+ ";";
                        }
                }
                map= ME.SM.OP.SM.AOP.MEC.SIQ.E.E_PLSQL_E.E_PLSQL(plsql, true);
        }catch(Exception e1) {
                //准备写回滚
                e1.printStackTrace();
        }
        return map;
    }
}
```

```
package OSV.ESU.standard;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import OSI.AOP.MEC.SIQ.plorm.PLORM_E;
import OSI.AOP.MEC.SIQ.plorm.PLORM_C;
//OSV PLSQL 研发，思想，设计：罗瑶光 20210507
```

```
//选中一个 cell 进行改变
public class U_CellStandard{
	public List<String> txtToList() throws IOException{
		List<String> dic_list= new ArrayList<>();
		return dic_list;
	}
	//   稍后我会把下面这个 2 个模式变成 ORM UNSQL，函数是很好写，但是我在思考
	//or and 太多 怎么进行规范传参..
	//   我的动机很简单，就是养疗经数据表可以直接数据库操作修改.
	//   见 DNA 编码与计算第一卷 308 页
	//    update samples
	//    tableName:test:update;
	//    condition:or:testCulumn1|<|20:testCulumn2|==|fire;
	//    condition:and:testCulumn1|>|100:testCulumn2|==|fire;
	//    culumnValue:date0:19850525;
	//    culumnValue:date1:19850526;
	//
	//    update samples tableName:test:update;
	//    condition:or:testCulumn1|<|20:testCulumn2|==|fire;
	//    condition:and:testCulumn1|>|100:testCulumn2|==|fire;
	//    join:backend:utest;
	//    condition:and:uCulumn3|<|20;
	//    relation:and:testCulumn1|==|uCulumn1:testCulumn2|!=|uCulumn2;
	//    culumnValue:date0:19850525;
	//    culumnValue:date1:19850526;

	public static Map<String, Object> UpdateCellORM(String tabKey, String rowId
			, String cellName
			, String cellValue) throws IOException{
		Map<String, Object> map= null;
		try {
			PLORM_C orm= new PLORM_E();
			orm.startAtRootDir("C:/DetaDB1").withBaseName("ZYY")
			.withTableUpdate(tabKey).withCondition("or")
			.let("ID").equalTo(rowId)
			.checkAndFixPlsqlGrammarErrors()//准备完善 plsql orm 语言 的语法检查函数 和修复函数。
			.checkAndFixSystemEnvironmentErrors()//准备完善 plsql orm 语言 的系统环境检查函数和修复函数。
			.withCulumnValue(cellName, cellValue)
			.finalE(true);
			//map= org.plsql.db.plsql.imp.E_PLSQLImp.E_PLORM(orm, true);
		}catch(Exception e1) {
			//准备写回滚
			e1.printStackTrace();
		}
		return map;
	}
	public static Map<String, Object> UpdateCellORMByRowId(String rootPath
			, String baseName, boolean unTest
			, String tabKey, String rowId, String cellName, String cellValue)
```

```
                    throws IOException{
        Map<String, Object> map= null;
        try {
            PLORM_C orm= new PLORM_E();
            orm.startAtRootDir(rootPath).withBaseName(baseName)
            .withTableUpdate(tabKey).withCondition("or")
            .let("ID").equalTo(rowId)
            .checkAndFixPlsqlGrammarErrors()//准备完善 plsql orm 语言 的语法检查函数 和修复函数。
            .checkAndFixSystemEnvironmentErrors()//准备完善 plsql orm 语言 的系统环境检查函数和修复函数。
            .withCulumnValue(cellName, cellValue)
            .finalE(unTest);
            //map= org.plsql.db.plsql.imp.E_PLSQLImp.E_PLORM(orm, true);
        }catch(Exception e1) {
            //准备写回滚
            e1.printStackTrace();
        }
        return map;
    }
    public static Map<String, Object> UpdateCellORMByEquals(String rootPath
            , String baseName
            , boolean unTest, String tabKey, String conditionSubject
            , String conditionObject
            , String cellName, String cellValue) throws IOException{
        Map<String, Object> map= null;
        try {
            PLORM_C orm= new PLORM_E();
            orm.startAtRootDir(rootPath).withBaseName(baseName)
            .withTableUpdate(tabKey).withCondition("or")
            .let(conditionSubject).equalTo(conditionObject)
            .checkAndFixPlsqlGrammarErrors()//准备完善 plsql orm 语言 的语法检查函数 和修复函数。
            .checkAndFixSystemEnvironmentErrors()//准备完善 plsql orm 语言 的系统环境检查函数和修复函数。
            .withCulumnValue(cellName, cellValue)
            .finalE(unTest);
            //map= org.plsql.db.plsql.imp.E_PLSQLImp.E_PLORM(orm, true);
        }catch(Exception e1) {
            //准备写回滚
            e1.printStackTrace();
        }
        return map;
    }
    //    update samples tableName:test:update;
    //    condition:or:testCulumn1|<|20:testCulumn2|==|fire;
    //    condition:and:testCulumn1|>|100:testCulumn2|==|fire;
    //    join:backend:utest;
    //    condition:and:uCulumn3|<|20;
    //    relation:and:testCulumn1|==|uCulumn1:testCulumn2|!=|uCulumn2;
    //    culumnValue:date0:19850525;
    //    culumnValue:date1:19850526;
    //    String plsql= "setRoot:C:/DetaDB1;" +
```

```
//     "baseName:ZYY;" +
//     "tableName:"+ tabKey+ ":update;" +
//     "condition:or:ID|==|rowId;" +
//     "culumnValue:cellName:cellValue;";
//    condition:"我似乎没有设计 rowid culumnid 的数字选项  函数,稍后补充下":19850526;
//    condition  可以用 uid in  或者  uid==  来实现.这样会导致计算变慢,所以  设计 rowid culumnid 的数字比较选项
//是有必要的.
    public static Map<String, Object> UpdateCellPLSQL(String tabKey, String rowId
            , String cellName
            , String cellValue) throws IOException{
        Map<String, Object> map= null;
        try {
            String plsql= "setRoot:C:/DetaDB1;" +
                    "baseName:ZYY;" +
                    "tableName:"+ tabKey+ ":update;" +
                    "condition:or:ID|==|"+ rowId+ ";" +
                    "culumnValue:"+cellName+ ":"+ cellValue+ ";";
            map= ME.SM.OP.SM.AOP.MEC.SIQ.E.E_PLSQL_E.E_PLSQL(plsql, true);
        }catch(Exception e1) {
            //准备写回滚
            e1.printStackTrace();
        }
        return map;
    }
}
```