1. **迁移步骤**
2. 定义数据库idl
3. 使用批处理生成orm映射h/cpp
4. 非从数据/从表，必须在SImplKVDBLoader::\_rpc\_call\_LoadHuman中手工增加载入代码
5. Human::Initial中会获得载入的数据，调用模块的函数，从HumanCollection中获得本模块的数据
6. 数据变更后，调用Human::GetOrmHelper()->OrmUpdate来保存
7. 玩家下线或切地图时需要保存的数据，在Human::FlushToOrmObjects中增加代码保存
8. **Idl说明**
9. 支持的数据类型
   1. Int32,int64,int16,uint8,bool 默认值为0或false
   2. string,定长string(有len参数)
   3. 自定义的结构
   4. 数组,哈希表(key为string)
   5. struct属性lightweight表示次结构为轻量级结构（不允需array,hashset等非内建或容器类型），内部将使用array保存次结构
   6. 值类型(生成的为值，不用指针，仅限普通struct及内建类型),次类型访问限制为public
   7. Hashmap 增加可选的key，可为int64或string，原语法<hashmap>扩展为<hashmap int64> 或 <hashmap string>
   8. Array,hashmap原来存的都是指针，现在接受<value\_type>修饰，例如生成的orm原来为pwutils::KvormArray<Human\*>，在接受<value\_type>修饰后，变为pwutils::KvormArray<Human>
   9. 删除Pwutils::KvormStaticArray 封装，如原来生成的的orm代码

Human humans <arrary> <len 3> 生成 KvormStaticArray<Human\*,3>，现在会生成 Human\* humans[3];，值类型也一样，由生成的KvormStaticArray<Human,3>变化为 Human humans[3]

示例：

struct Element

{

Int16 field;  
}

struct Test

{

uint8 field1 < default 1>;

string field2; // 变长字符串

string field3 <len 10>; // 定长字符串

Element field4 <struct>; // 自定义结构

Element field5 <array> <struct>; // 自定义结构数组

Element field6 <array> <struct> <len 3>; // 自定义结构定长数组

Element field7 <hashmap> <struct>; // hash表 string->Element\*

Element field8 <value\_type> <struct>; // 值类型，生成非指针成员变量  
}

1. 结构属性
   1. <slave Master> 表示本结构从属于Master结构，Master与本结构为一对一关系，用masterid关联，使用了这个属性，结构体内必须为int64 | string且含masterid属性的字段
   2. <slave\_table Master>表示本结构从属于Master结构，Master与本结构为一对多关系，用masterid关联，使用了这个属性，结构体内必须为int64 | string且含masterid属性的字段，以及一个含key属性的字段
2. 字段属性
   1. <key [index]> 结构主键，在有多个key的情况下，由index决定组合顺序，如 int64 v1 <key 2>; int64 v2 <key 1>;组合主键为 v2.v1
   2. <oldname “旧字段名”> 在字段有新名字时，必须注明旧名字，以保证数据载入，且这个字段名永远不能再使用。本属性可有多个
   3. <len Number> 指定string或数组的长度，如果是string数组的话，只修饰数组
   4. <array> 标明本字段为数组
   5. <hashmap> 标明本字段是哈希表
   6. <struct> 标明本字段是自定义结构
   7. <bitmask> 标明本字段是掩码，自动生成include/exclude等函数
   8. <count> 标明本字段是计数器，自动生成plus,minus函数
   9. <value\_type>标明本字段是直接值类型
   10. <default N | “str”> 标识本字段的默认值,对array,hasmmap无效
3. **自增id池**

除创建角色的自增id直接从数据库获取外，其它运行中的id均由id池生成

如何增加一个id池

* + 1. 在pwutils:: KvormUUIDPool中增加类型TYPE\_ITEM的枚举
    2. 在InitialWorld中调用CreateGenerator创建id生成器
    3. 调用KvormUUIDGenerator的Spawn函数获得id，调用Release释放id

1. **陷阱**
   1. 为节约空间，默认值不会保存，意味着如果在一个字段中用 default定义了新的默认值，已保存的数据中，只要没有改过值，那载进来，就会变成新的指定值
   2. 除非结构所有字段都是value\_type，不在该结构生成的orm类不能复制，只能通过clone来建创新对象
   3. 所有自动生成的orm类，都不能用memset清除，或者memcpy复制，这样会导致vptr失效，调用函数将导致内存错误，如果需要传送到其它进程，应收到次结构时，使用orm::Test t = \*(orm::Test\*)chunk.buf来复制一下，保证vptr正确
   4. orm类对象，在保存增加hash比较，如果没有变更，不会写到数据库，带来一个陷阱，使用from\_bson复制一个对象obj后，直接OrmHelper:GOrmUpdate(obj)，不会保存，因为obj没有变化，必须加一句obj->hash\_update(0)后才可以保存
   5. 使用lightweight修饰的结构，字段必须使用index来保证结构的成员与bson数组对应
2. **其它**
   1. pwcommon抽成一个单独的静态库，生成的orm代码放在这个项目的kvorm目录中
   2. 有被其它struct使用slave,slave\_table引用的结构，会自动生成collection代码，用于管理,组织数据
   3. Item结构不从属任何结构，因此载入时多了一重间接性
   4. 新的数据库线程为
      1. SImplKVDB 异步 port=cst\_port\_dbsrv\_kv\_service
      2. SImplKVDBLoader 同步载入 port=cst\_port\_dbsrv\_kv\_service\_loader