docxit 部分函数和协议设计

目录

.docxitPath 文件格式

该文件保存在 `/home/<username>/` 下，由 `docxit init` 创建，记录该用户所有 docxit 仓库的完整路径名，用于辨识当前位于哪个仓库中。

```
/path/to/your/docxit/repository1/
/path/to/your/docxit/repository2/
...
/path/to/your/docxit/repositoryn/
```

index 文件格式

暂时先这样，以后可能考虑压缩存储空间。暂时不做重命名类型。暂时不考虑文件权限问题。

该文件保存了暂存区域整个目录树的信息，记录从项目初始化到目前为止，项目仓库中所有文件文件名、sha1值等

记录格式

```
typedef enum{
    add, remove, changed, unchanged
}DocxitRecordKind;

typedef struct{
    DocxitRecordKind kind;
    char name[4096];        // 相对于 docxit 管理的根目录的路径名
    char key[40];           // 记录旧 key
    char newkey[40];        // 记录新 key，在 add、unchanged 或 remove 中没有意义
}DocxitRecord;
```

文件格式

```
n 的字符串形式<LF>
n 个 DocxitRecord 的二进制存储
```

注：可以考虑从固定字节处开始存放二进制的 DocxitRecord 以方便 fseek。可能需要考虑结构体对齐

对象格式

BLOB 对象格式

直接存储二进制的 docx，暂时不压缩

tree 对象格式

暂时定为

```
m n
<filename1> <key1>
<filename2> <key2>
...
<filenamem> <keym>
<dirname1> <key1>
<dirname2> <key2>
...
<dirnamen> <keyn>
```

commit 对象格式

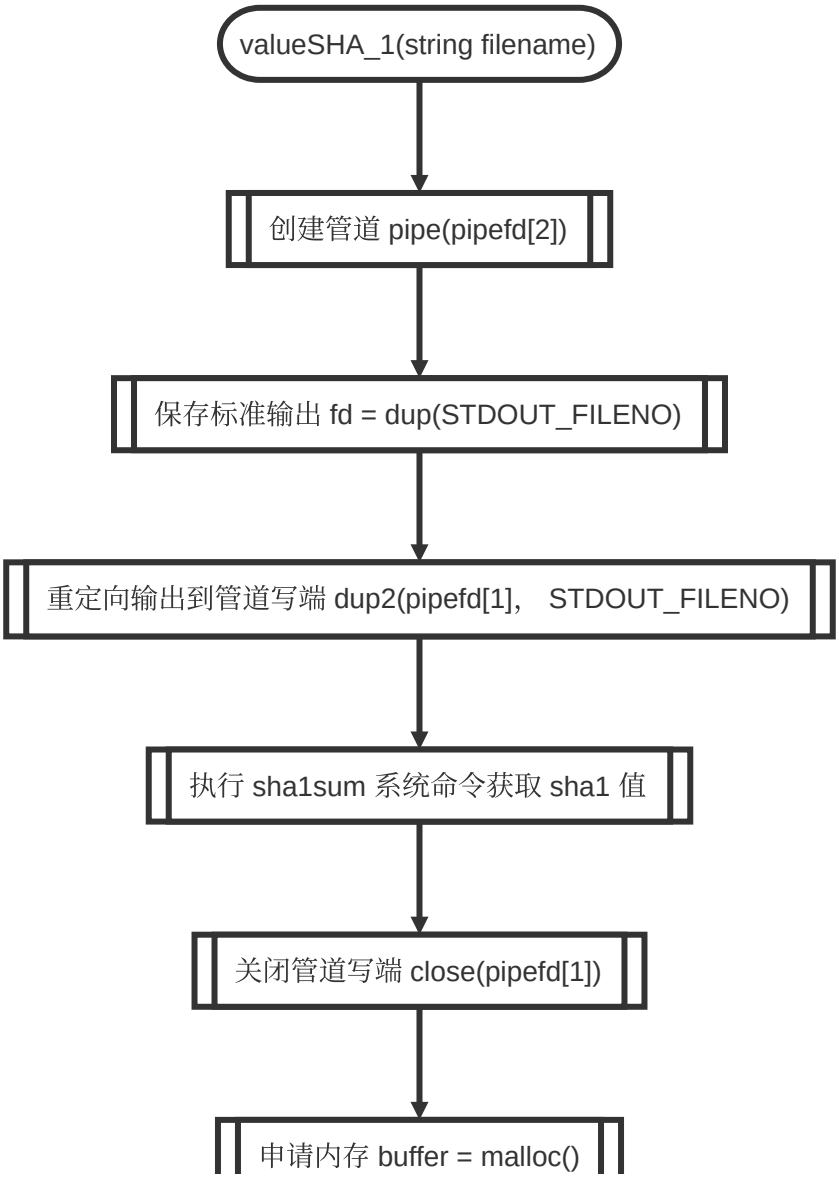
暂时定为

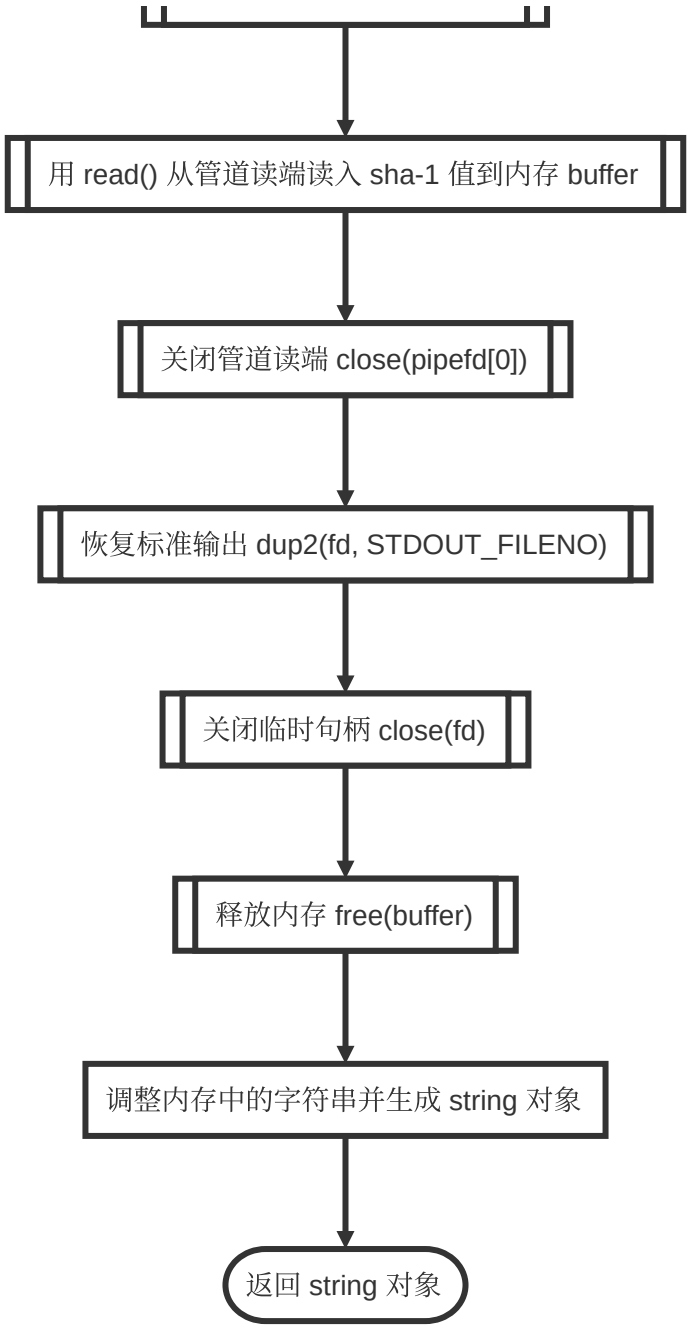
```
<tree_root_key>
<author_info>
<date>
<commit_message>
<parent_commit_object_key>
n
<child_commit_object_key1>
<child_commit_object_key2>
...
<child_commit_object_keyn>
```

底层函数

valueSHA_1

- 参数是文件相对于版本根文件夹的相对路径名，不包括根文件夹名
- 返回的 string 给出该文件的 sha-1 值，40 个字符。

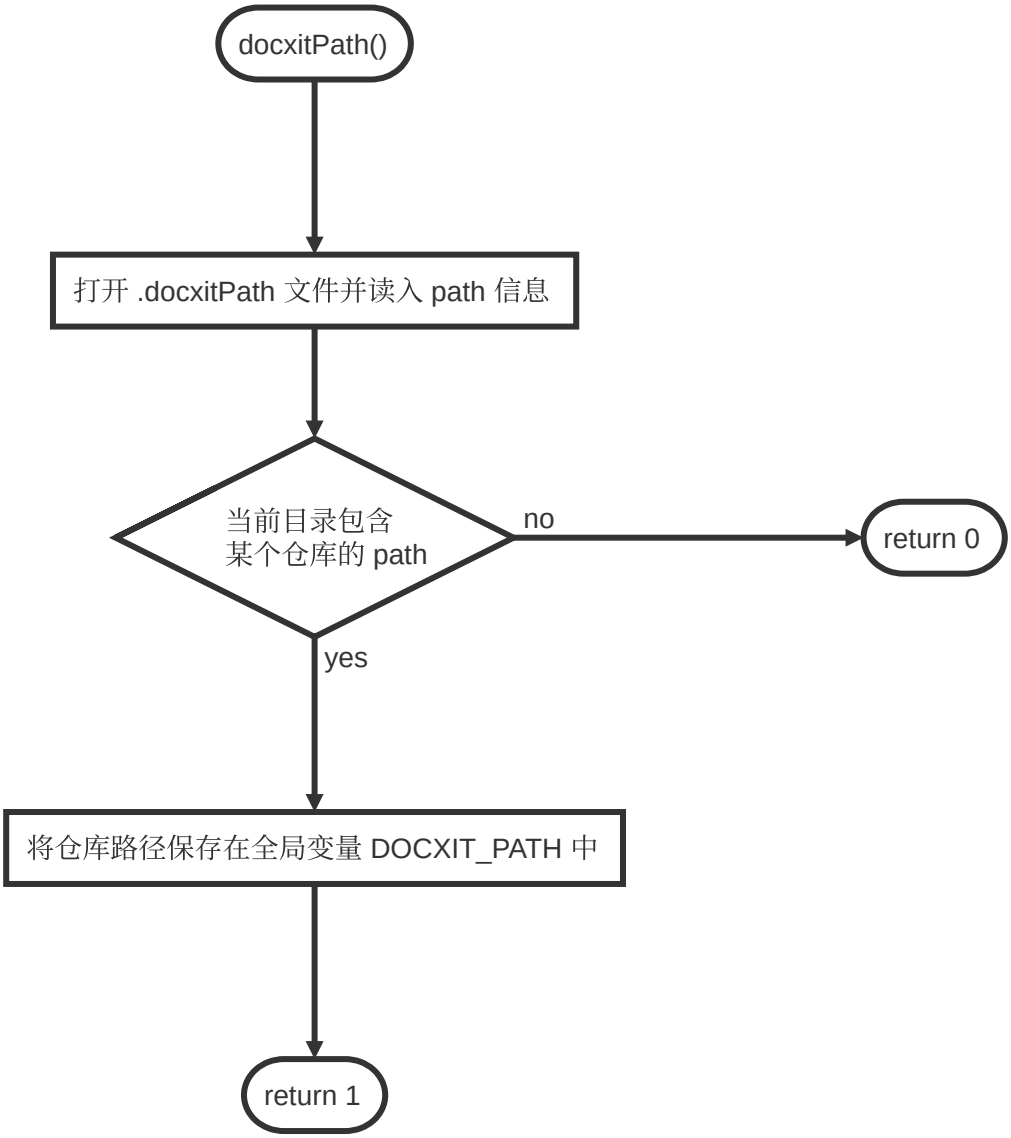




docxitPath

暂时不支持仓库嵌套

- 访问 .docxitPath 文件，找到当前目录所在的仓库，将路径赋值给全局变量。
- 返回 1 表示成功，0 表示失败。



docxit add

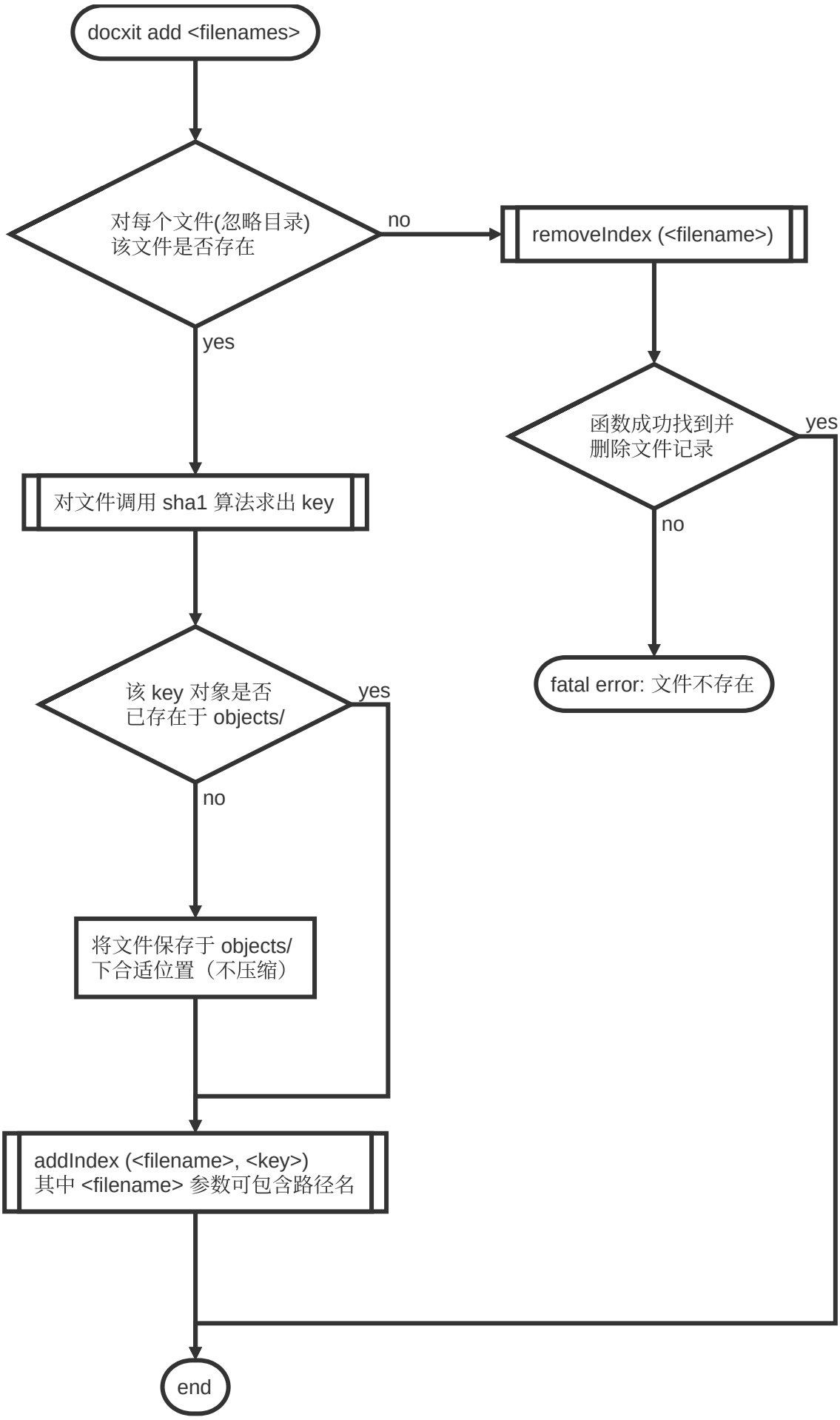
暂时不实现 rm，因为可以通过 add 达到同样的效果

*我们要求添加文件可以用. .. ? * 但删除文件必须全部 add 进来*

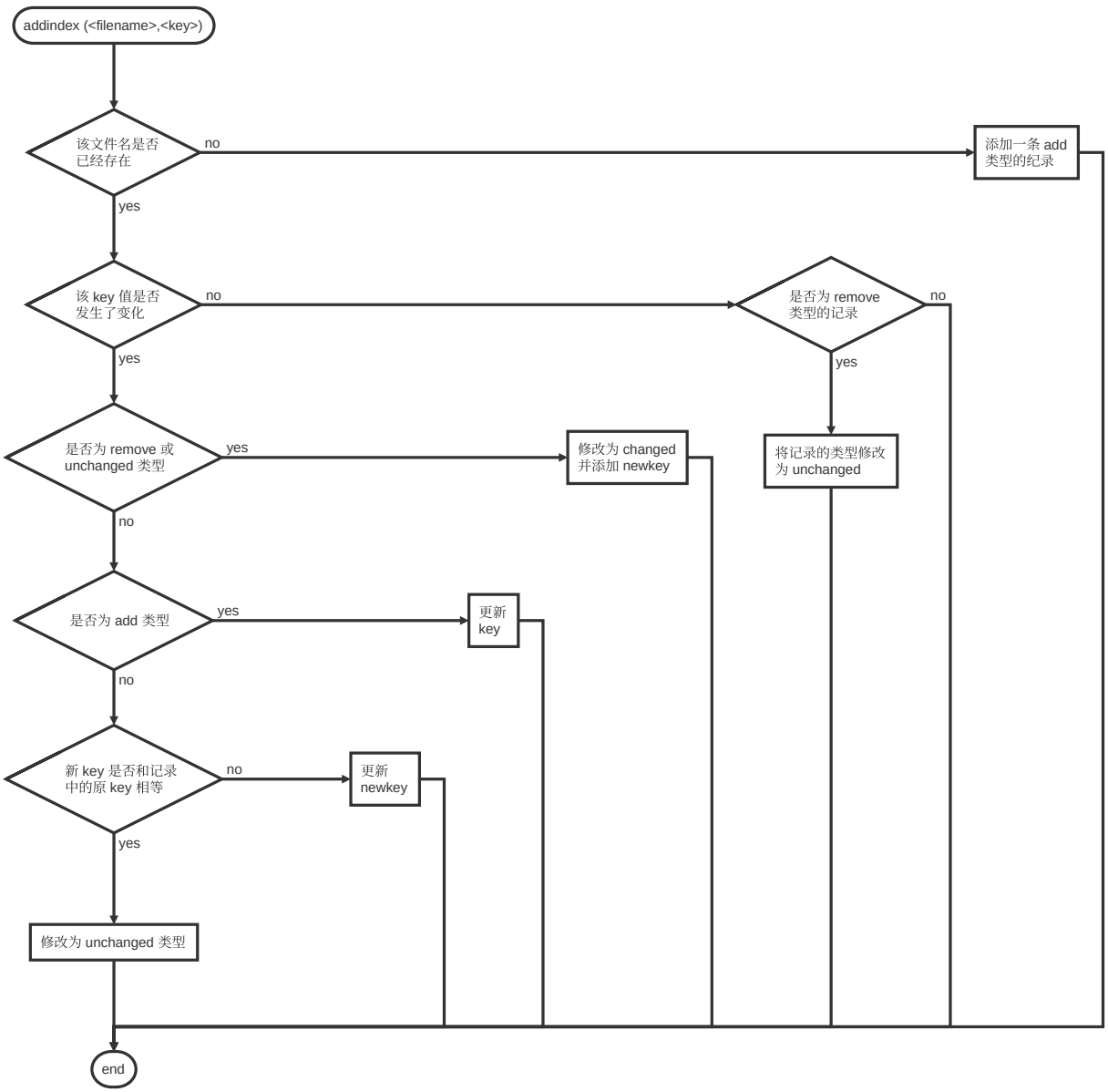
主要任务

- 获取文件 sha-1 值并将文件保存为 BLOB 对象
- 将文件信息记录到 index 文件

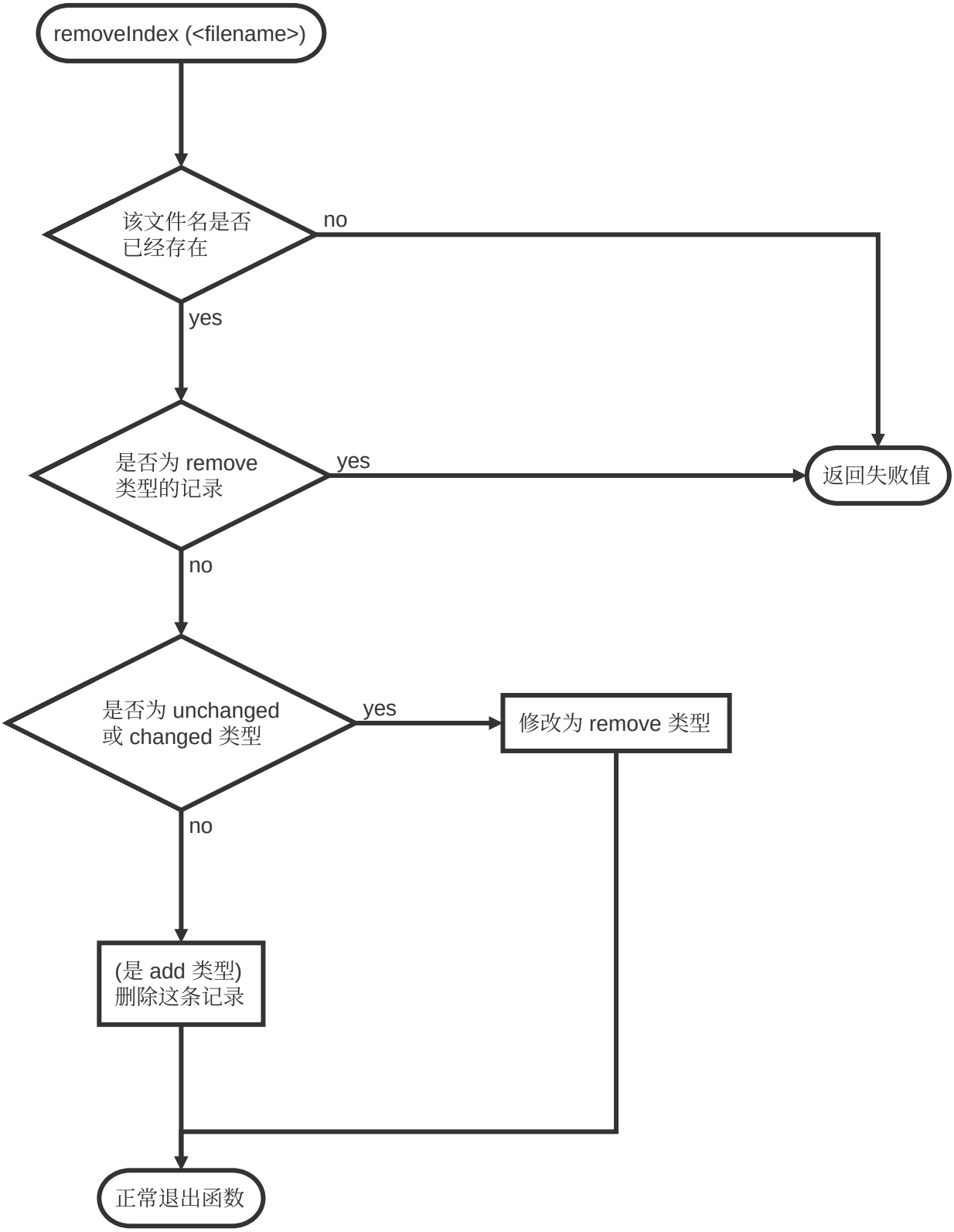
主函数


docxit add <filenames>
对每个文件(忽略目录)
该文件是否存在
no
removeIndex (<filename>)
yes
函数成功找到并
删除文件记录
yes
no
fatal error: 文件不存在
对文件调用 sha1 算法求出 key
该 key 对象是否
已存在于 objects/
yes
no
将文件保存于 objects/
下合适位置（不压缩）
addIndex (<filename>, <key>)
其中 <filename> 参数可包含路径名
end

addIndex


addindex (<filename>,<key>)
该文件名是否已经存在
no
添加一条 add 类型的纪录
yes
该 key 值是否发生了变化
no
是否为 remove 类型的记录
no
yes
yes
是否为 remove 或 unchanged 类型
yes
修改为 changed 并添加 newkey
将记录的类型修改为 unchanged
no
是否为 add 类型
yes
更新 key
no
新 key 是否和记录中的原 key 相等
no
更新 newkey
yes
修改为 unchanged 类型
end

removeIndex

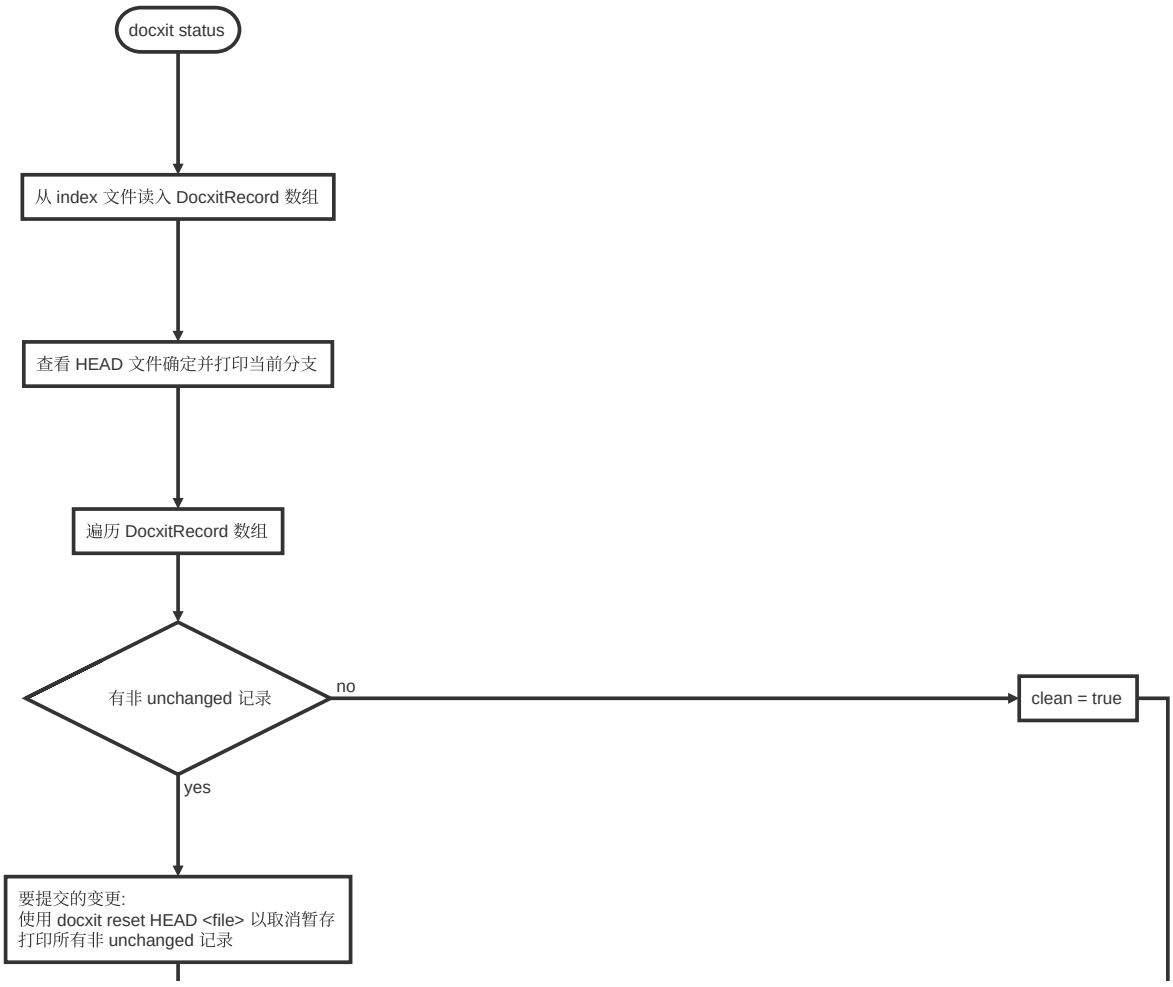


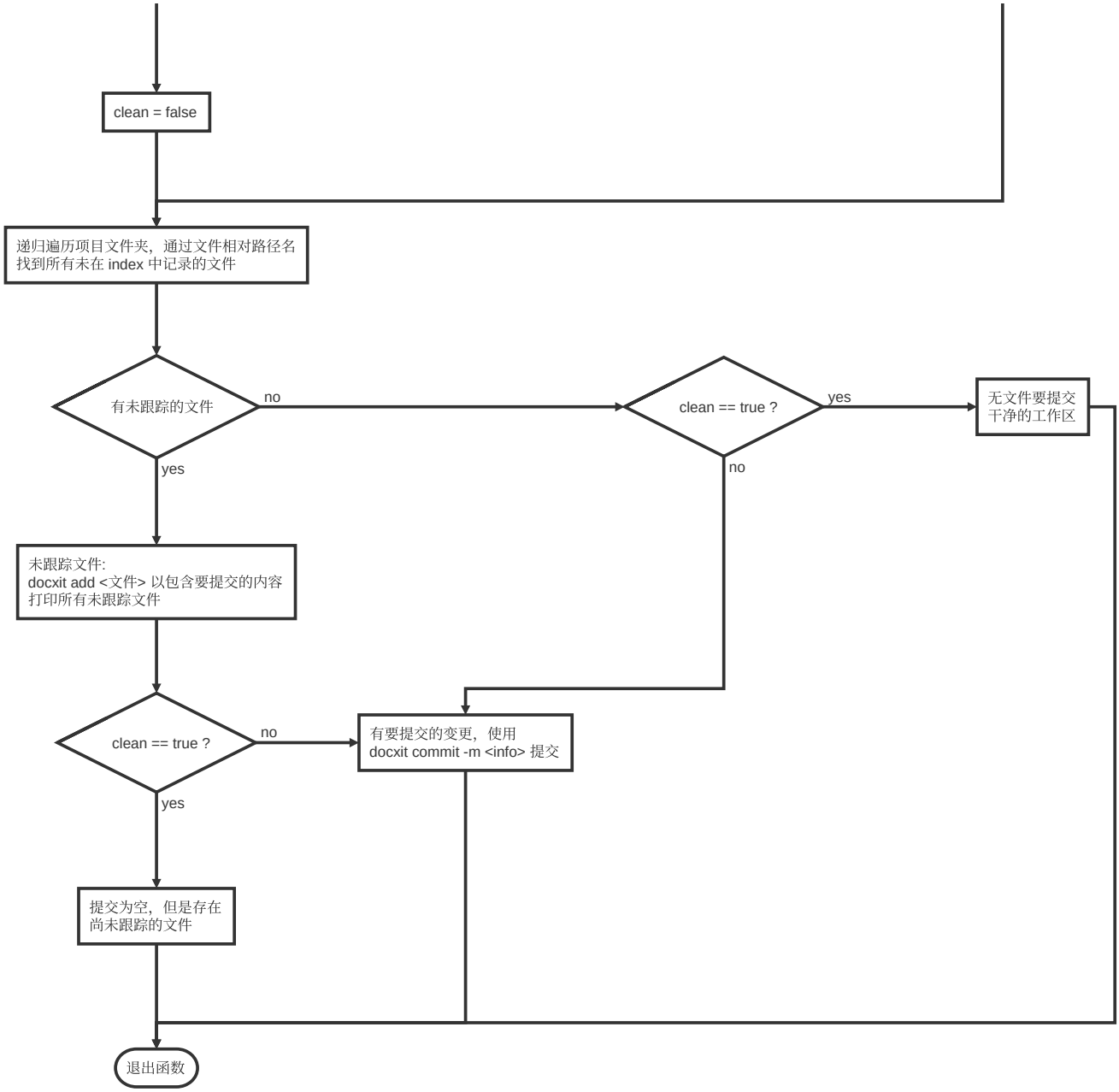
docxit status

主要任务:

- 根据 index 文件确定是否有要提交的文件并输出相应信息
- 根据工作区情况确定是否有未跟踪的文件并输出相应信息

主函数




clean = false
递归遍历项目文件夹，通过文件相对路径名
找到所有未在 index 中记录的文件
有未跟踪的文件
no
clean == true ?
yes
无文件要提交
干净的工作区
yes
no
未跟踪文件:
docxit add <文件> 以包含要提交的内容
打印所有未跟踪文件
clean == true ?
no
有要提交的变更，使用
docxit commit -m <info> 提交
yes
提交为空，但是存在
尚未跟踪的文件
退出函数

注：整个文件夹未记录时，可将整个文件夹作为未跟踪的文件

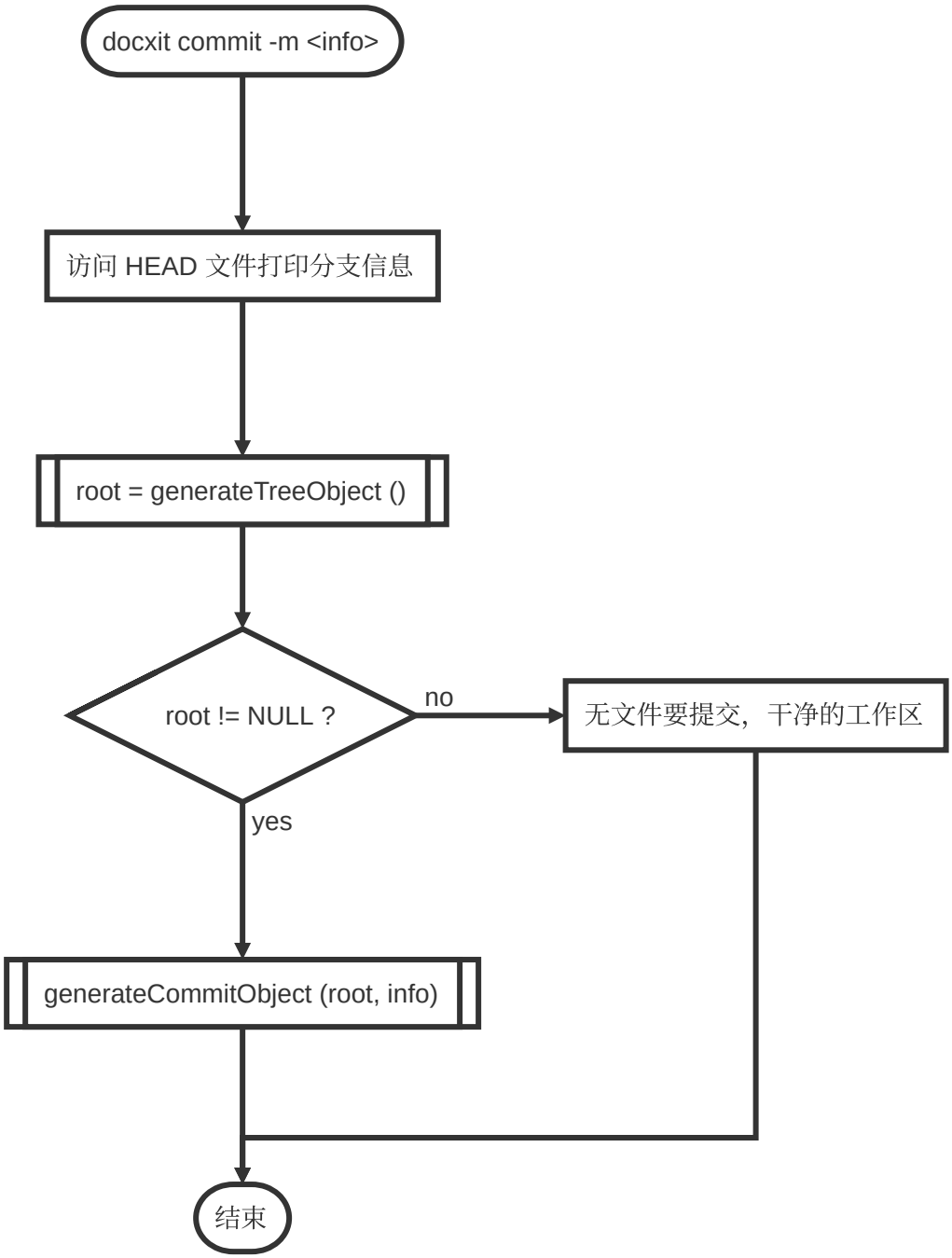
直接存储这个 index 文件作为一个版本太浪费空间，比如你的项目有一个 a 文件夹，里面有 100 个文件，如果有 10 个版本，每个版本都需要记录这 100 个文件，这需要 1000 条记录；而如果把这个文件夹抽象成 tree 对象（其中存有 100 条记录），在 10 个根 tree 对象每个只需要存储 1 个这个 tree 对象（即记录了这 100 个文件），只需要 100 + 10 = 110 条记录，节省了接近 90% 的空间占用。

docxit commit

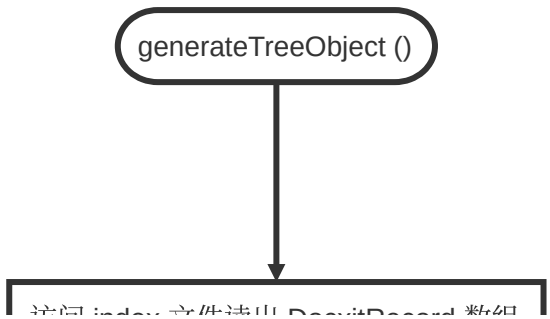
主要任务：

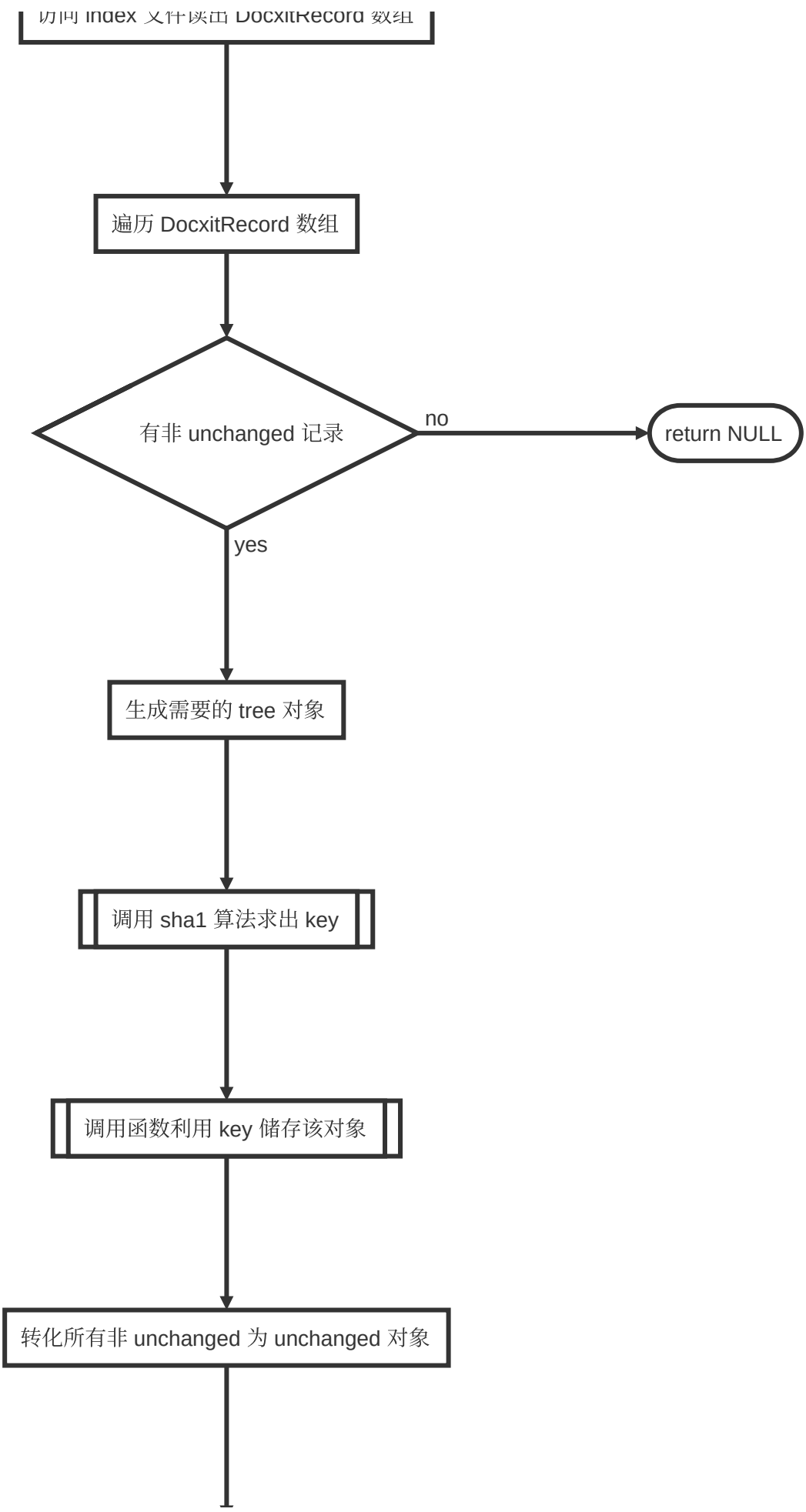
- 由 index 文件生成一个根 tree 对象以及其他所有需要的 tree 对象
- 生成 commit 对象并记录根 tree对象指针和提交信息

主函数



generateTreeObject




访问 index 文件读出 DocxitRecord 数组
遍历 DocxitRecord 数组
有非 unchanged 记录
no
return NULL
yes
生成需要的 tree 对象
调用 sha1 算法求出 key
调用函数利用 key 储存该对象
转化所有非 unchanged 为 unchanged 对象

generateCommitObject

暂定提交树用双向链表（树）表示，即树的双亲孩子表示法