系统管理(权限系统)操作手册

1. 权限系统操作

权限系统操作手册：

在系统管理菜单内进行以下操作。

2. 机构/用户/角色管理/用户授权

①创建用户前至少要创建一个机构,再在用户管理中创建用户

②角色可在用户新增/编辑时修改，多对多授权可在用户授权中赋权，单个角色下添加用户可在角色管理-用户列表-选择用户处添加。

2.1 用户管理

①解除锁定：是指用户登录账户密码错误超过指定次数后，处于禁用锁定状态无法登录，需要管理员手动解锁。

②停用：用户若被停用，则无法登录使用系统，停用后可再手动启用。

注：在禁用锁定状态前有一个错误锁定状态，若处于错误锁定状态，则会在达到系统设定的锁定周期后自动解锁，不需要管理员手动解锁，两种状态的锁定次数皆可在密码策略中由管理员设置

2.2 角色管理

① 角色权限保存：若左侧角色列表无勾选项，则保存右侧选择的权限至当前点击行的角色；若左侧勾选一或多个角色，则保存右侧选择的权限至勾选的全部角色中。（用户授权保存同）。

②角色的复制粘贴：点击复制，复制该行所拥有的各类权限(不包括用户列表);再点击需要复制的行查看复制前的权限(此步骤为了方便查看复制前后效果对比，可省略，直接粘贴)，点击粘贴可看到权限变化。

注：粘贴后需要保存方能生效，若在保存前点击或查看其他行，则不保留复制效果

③ 机构权限与用户列表： 用户列表处可用来添加/移除当前选择角色下的用户。

注：一是在添加用户前必须确保机构权限已选，只能添加已有权限机构下的用户；二是添加/移除需要统一保存后方可生效,三是需要配置按钮权限需要修改前端按钮的代码（详情参照前端按钮权限控制的文档）

2.3 用户授权

用户授权：主要用于将角色分配给具体用户，可多对多。

3. 数据权限

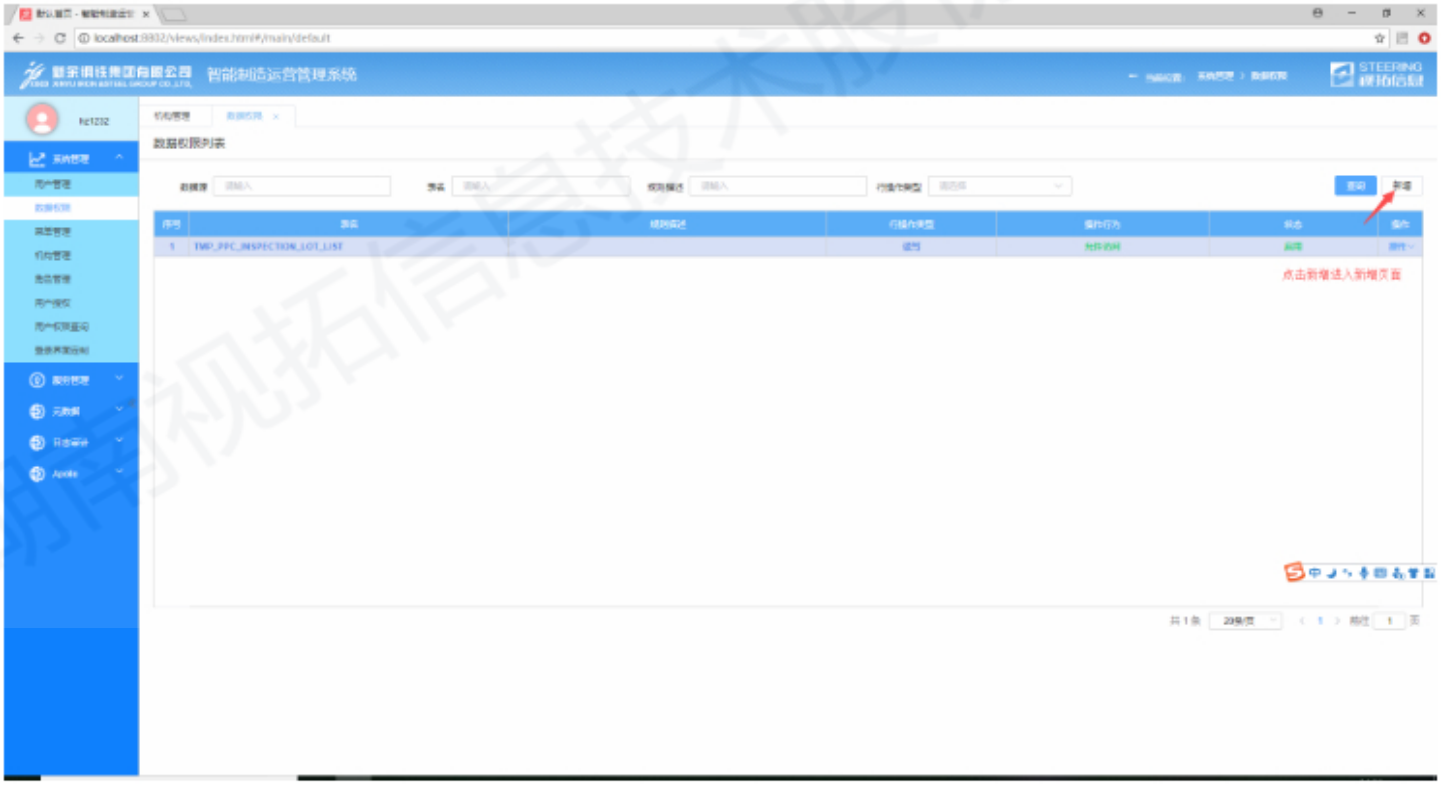
数据权限设置是针对元数据库表相关访问的限制设置，分为行权限设置和列权限设置

数据权限操作的库表基于基础服务的元数据管理（包含系统、库表、字段、外键）。

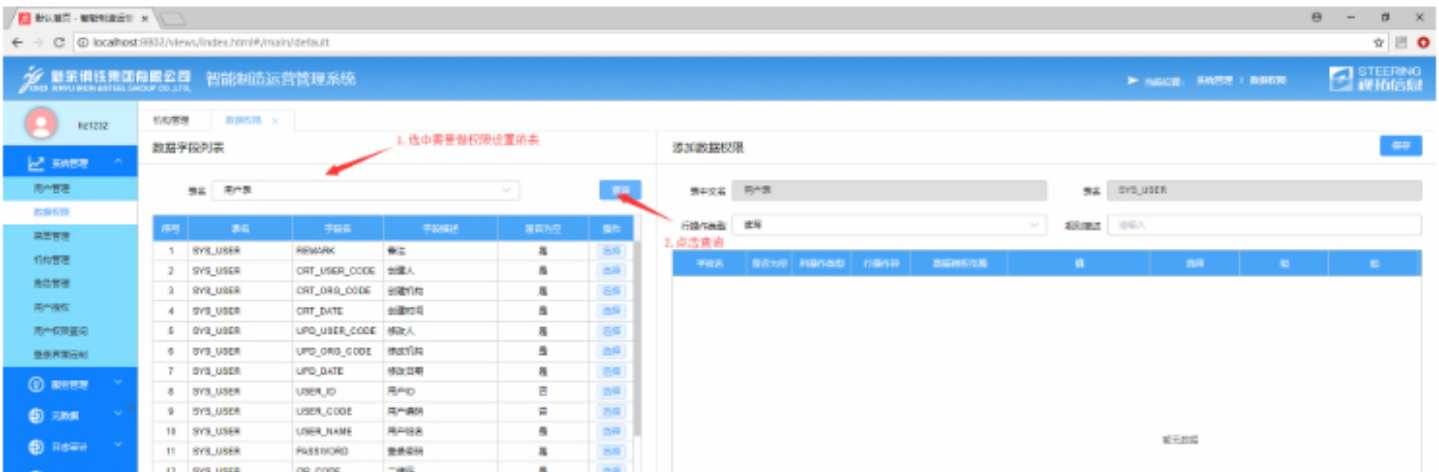
行权限设置：设置行权限相当于执行sql的时候时候加上一个WHERE条件列权限设置：通过查询某一列的时候返回结果为空

3.1 行权限设置

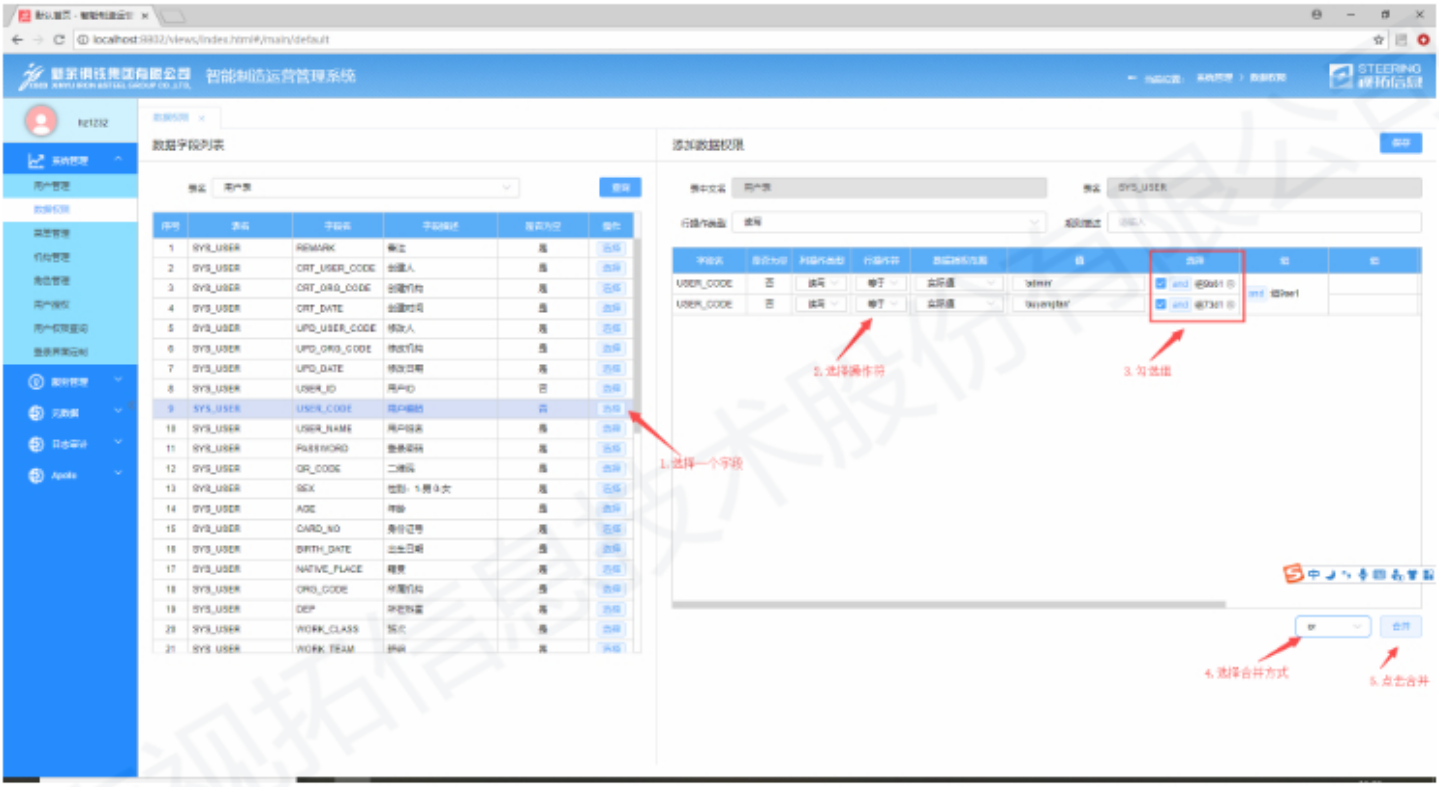
3.1.1 进入权限新增页面



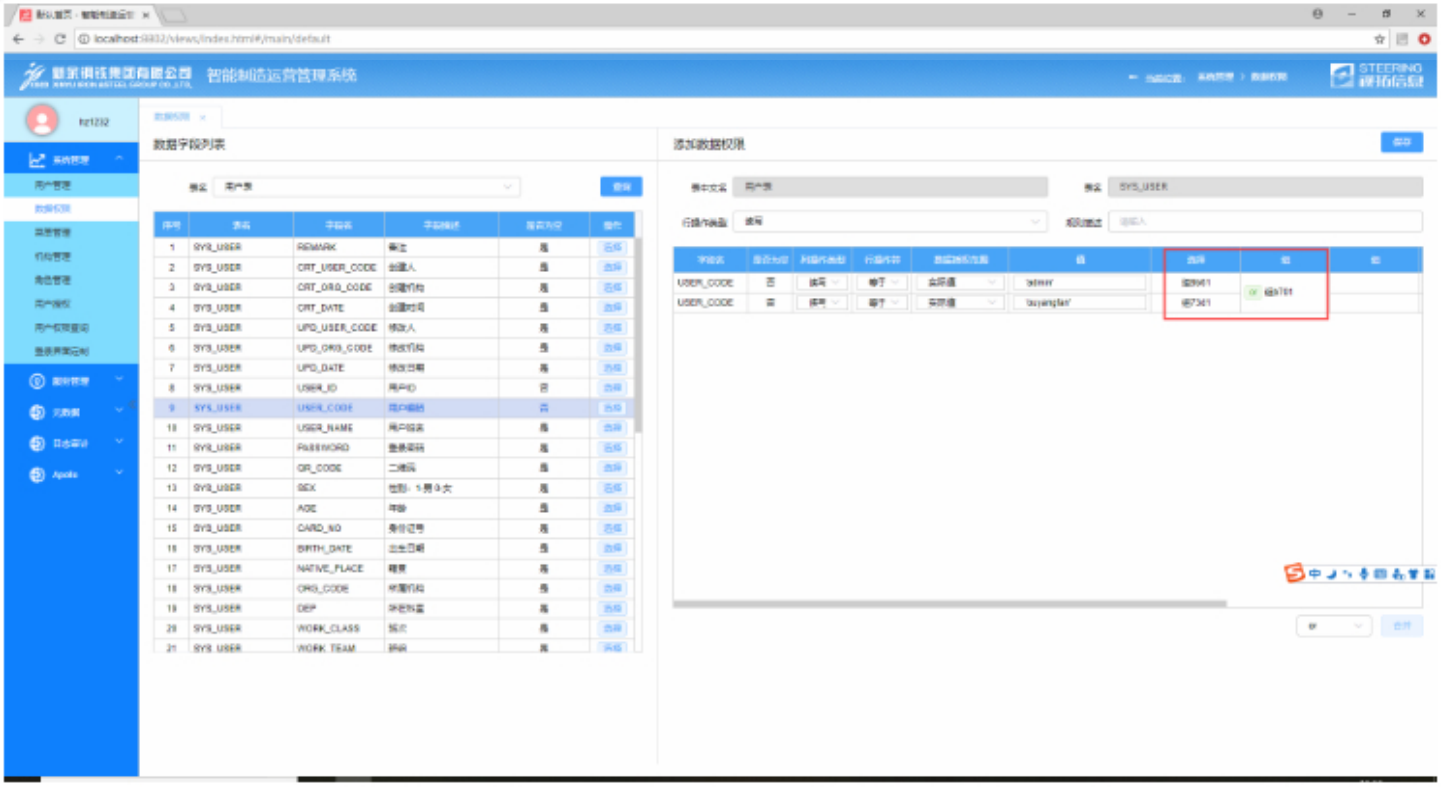
3.1.2 选择需要做权限控制的表



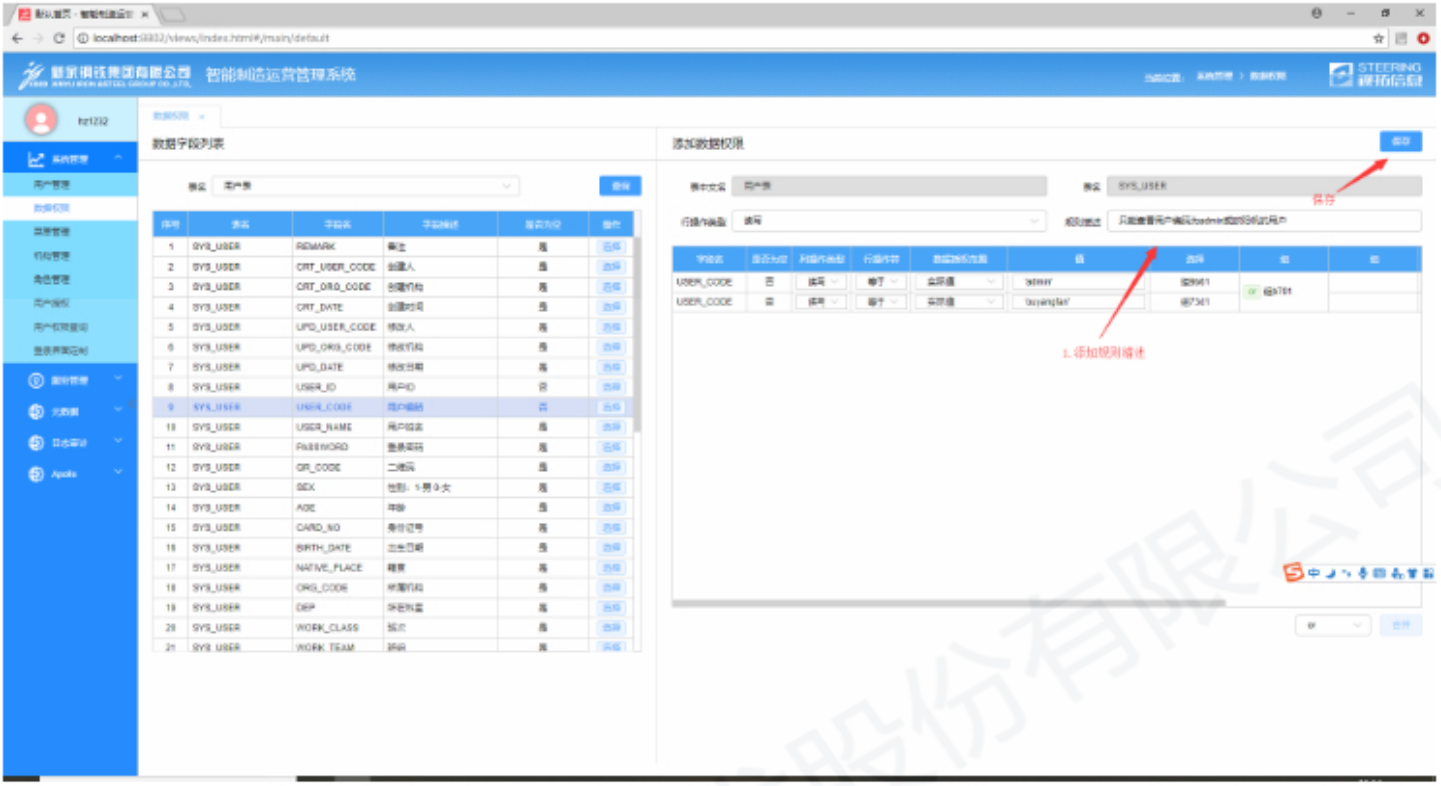
3.1.3 添加一条数据权限



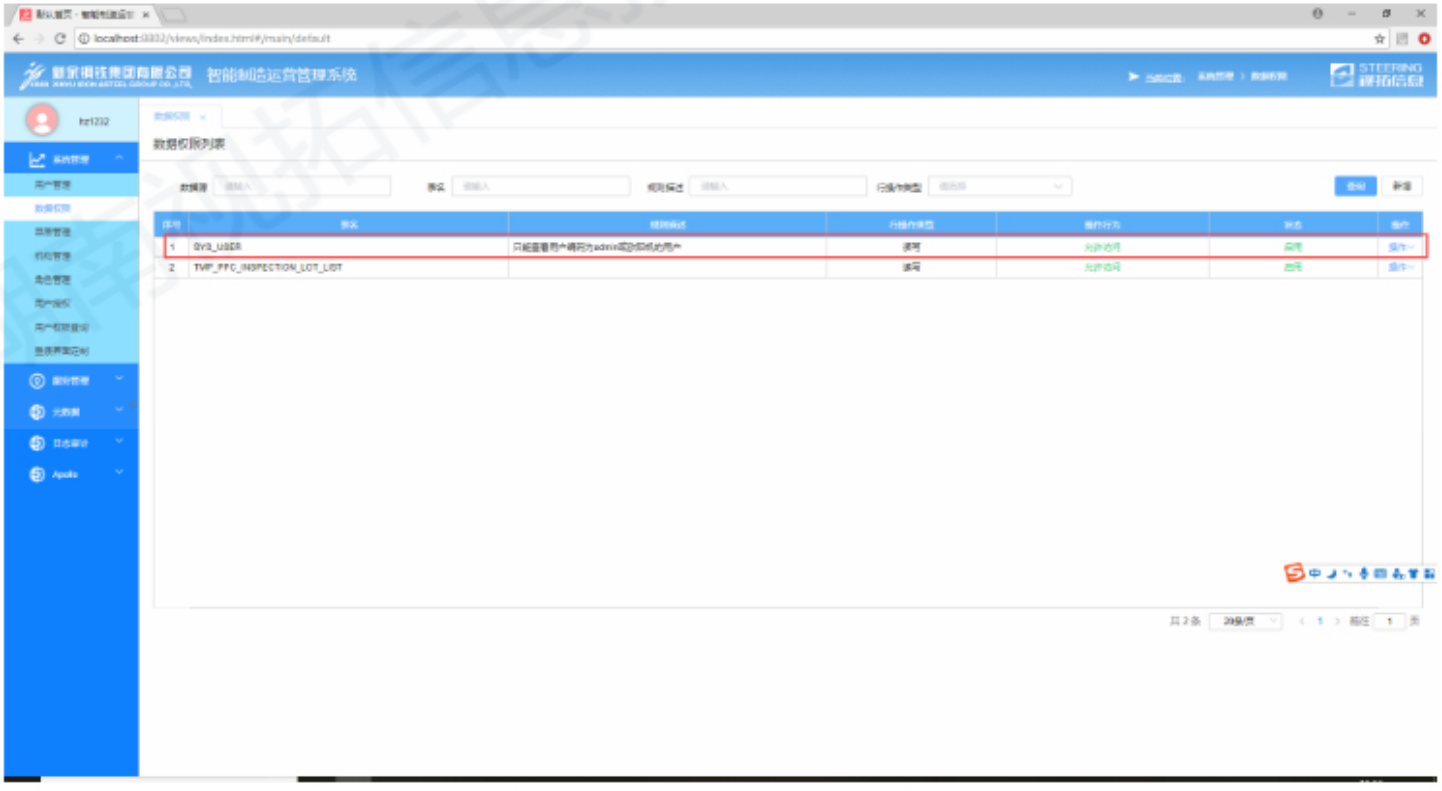
3.1.4 点击合并后的效果



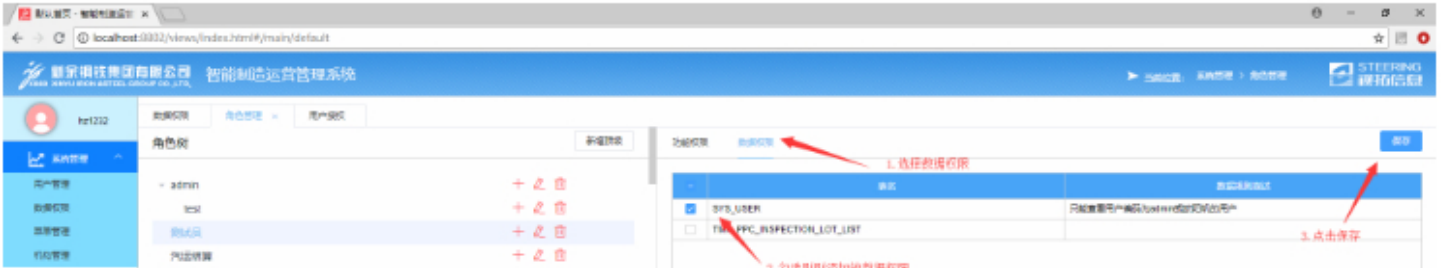
3.1.5 添加规则描述并保存



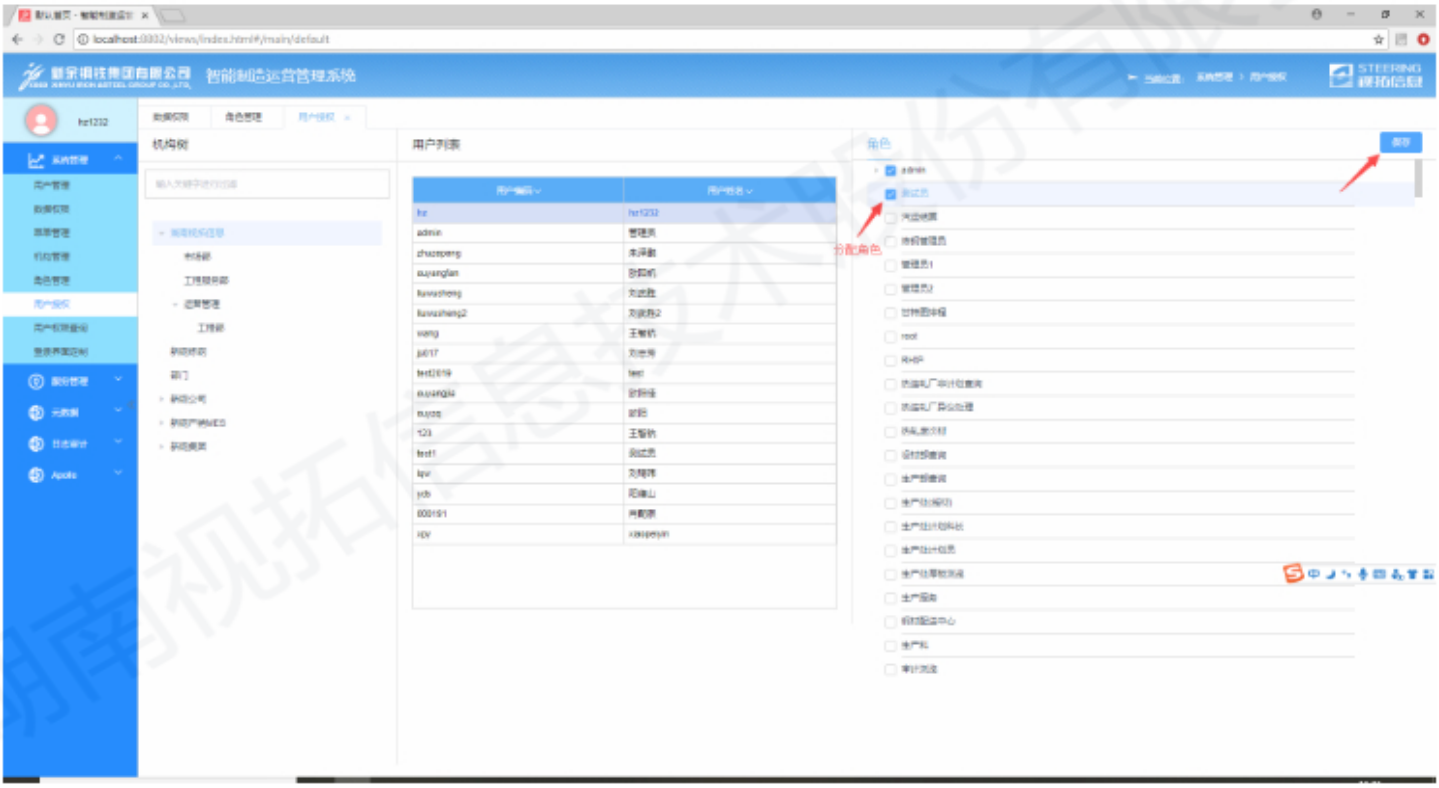
3.1.6 保存后可以看到刚刚添加的数据权限



3.1.7 给测试员角色赋权



3.1.8 给hz用户分配测试员角色

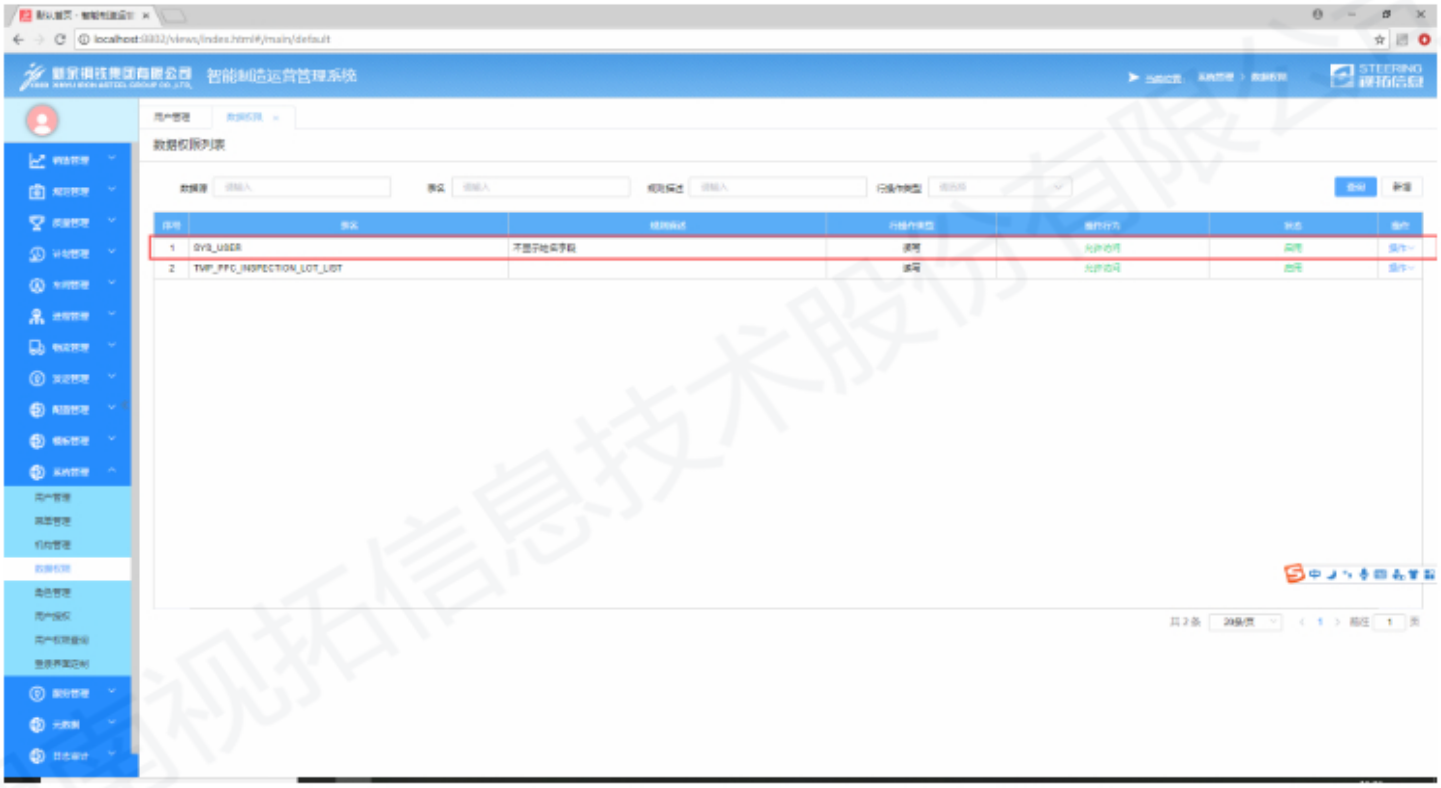


3.2 列权限设置

创建规则几步参考行权限设置

3.2.1 添加一条列规则

3.2.2 查看列规则



4.元数据管理

元数据管理是针对数据权限的系统库表以及其字段相关的数据管理。

数据权限的所有操作基于元数据的数据库关系，其管理操作与数据库结构相一致，元数据管理页面关联如下：

1、库表、字段、关联管理页面都默认展示所有数据。当从其他页面进入时则显示勾选项关联数据。

2、新增应用系统，在应用系统下建立多个表。

3、在库表管理中勾选某条数据进入：

点击关联按钮则进入外键管理页面，查看与其相关的关联表联系;

点击字段按钮进入字段管理查看该表下已有的字段信息;

点击导出表可导出勾选的当前表信息，导出系统则导出勾选数据所属系统下所有表的数据。

5.编码管理

编码管理是为数据表中某一具体字段自定义编码排列规则

① 点击新增按钮（如需修改可点击该行），在下方编辑区添加完成后，在table点击该行编码规则按钮可进入编码规则配置页面。

（编码规则名为：表名 + 字段名组成，其值唯一，意为表中的每个字段都只可对应一条编码规则）

② 进入规则配置页面后，点击添加规则

根据需要选择性添加并自定义规则类型，除固定值可重复添加外，其他都只可添加一次

可通过修改排列顺序调整该规则值在编码中的位置

停用状态的规则在整体编码中不生效

③ 添加完成后点击测试，验证编码效果

④ 接口引用

根据编码规则名查询详细规则数据

```
v1/syscodepatterns/queryByBuilderType?builderType=编码规则名
```

筛选上一步res.data中status===‘1’的数据，存为codeRuleList

获取业务编码个例 /v1/syscodebuilders/generateOneCode?builderType=编码规则, { codeRuleList: codeRuleList }

前端利用此接口在添加时引入获取的编码个例

6. 数据字典

新增/编辑数据字典，新增操作由开发人员建立并记住字典类型，在其他需要引用的界面调用接口获取所需字典类型： v1/sysdicts/listDict?pDictId=ROLE_TYPE&status=1。

其中pDictId的值为数据字典中的字典类型值，status=1代表启用状态；

7. 公告管理

是为了方便管理员针对不同角色发送实时通知：当用户登录时可实时接收公告通知而不必再次刷新页面。

8. 密码策略

密码策略是针对用户密码的配置规则，主要包括系统的初始密码设定，用户修改密码需满足的规则，用户创建时用户编码的限制条件，登陆错误锁定/密码禁用锁定次数,错误锁定的时间周期。

9. 数据备份

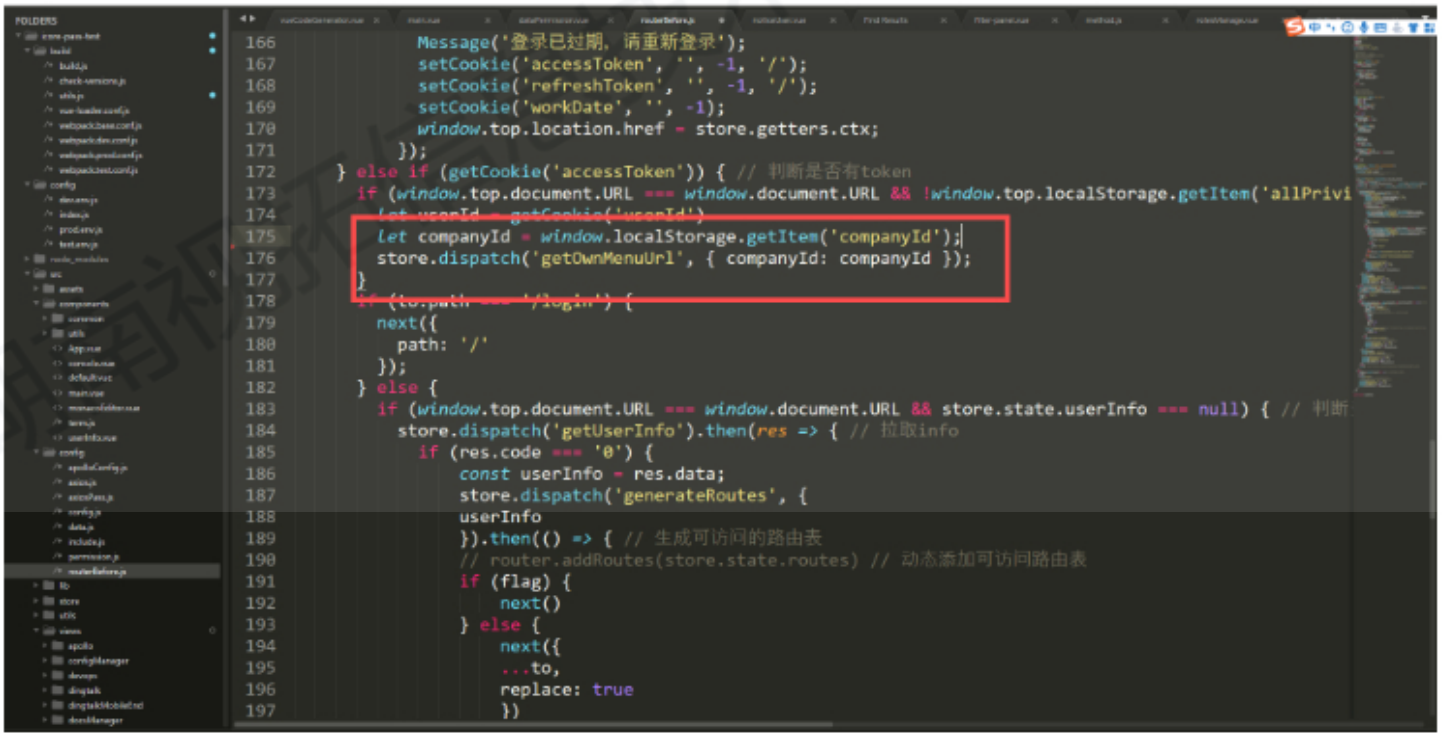
针对用户/角色/机构/菜单等数据作出备份，每条备份只可还原一次。 主要在于用户在用户/角色/机构/菜单等数据才做中出现错误或者想要回退数据所使用

注：长时间的备份记录还原时可能会失效，不要过于依赖此功能。

10. 权限改造(相对老版本的对比说明)

菜单权限

不再登录后全部存入，改为进入页面前获取，同样需在routeBefore.js中同步修改



```
166              Message('登录已过期，请重新登录');
167              setCookie('accessToken', '', -1, '/');
168              setCookie('refreshToken', '', -1, '/');
169              setCookie('workDate', '', -1);
170              window.top.location.href = store.getters.ctx;
171            });
172        } else if (getCookie('accessToken')) { // 判断是否有token
173          if (window.top.document.URL === window.document.URL && !window.top.localStorage.getItem('allPrivi
174            let userId = getCookie('userId')
175            let companyId = window.localStorage.getItem('companyId');
176            store.dispatch('getOwnMenuUrl', { companyId: companyId });
177          }
178          if (to.path === '/login') {
179            next({
180              path: '/'
181            });
182          } else {
183            if (window.top.document.URL === window.document.URL && store.state.userInfo === null) { // 判断
184              store.dispatch('getUserInfo').then(res => { // 拉取info
185                if (res.code === '0') {
186                    const userInfo = res.data;
187                    store.dispatch('generateRoutes', {
188                    userInfo
189                    }).then(() => { // 生成可访问的路由表
190                    // router.addRoutes(store.state.routes) // 动态添加可访问路由表
191                    if (flag) {
192                        next()
193                    } else {
194                        next({
195                        ...to,
196                        replace: true
197                        })
```

src-store-index.js文件中获取权限数据方法改动如下

```js
getOwnMenuUrl ({
    commit,
    state
  }, obj) {
    obj = JSON.parse(JSON.stringify(obj));
    let companyId = window.localStorage.getItem('companyId') ? window.localStorage.getItem('comp
    let res = (obj && obj.companyId) ? axios.get(proPath + 'v1/sysmenus/findFunctionMenusByUserI
    // let res = (obj && obj.userId && obj.menuId) ? axios.get(proPath + 'v1/sysmenus/findFuncBy
    res.then(res => {
```

```
      window.top.localStorage.setItem('ownPrivilege', JSON.stringify(res.data));
      //   放入东西
      commit('refreshOwnPrivilege', res.data);
    });
    return res;
  }
```

按钮权限 不再登录后全部存入，改为进入页面前获取，同样需在method.js中同步修改指令方法(代码部分如下)，本操作兼容之前版本，之前的页面无需修改，仍在按钮在调用v-privilege指令

```js
// 新的权限指令封装
Vue.directive('privilege', {
  inserted: function (_el, _binding) {
    let menuId = window.activeMenu || window.localStorage.getItem('activeMenu');
    let checkFoo = (privilegeData, {el, binding} = {el: _el, binding: _binding}) => {
      let privilegeFlag = false;
      let privilegeMark = binding.value;
      // console.log({el, binding})
      for (let i = 0; i < privilegeData.length; i++) {
        if (privilegeData[i].privilege === privilegeMark || (menuId + privilegeData[i].privilege
          privilegeFlag = true;
          break;
        }
      }
      if (!privilegeFlag) {
        $(el).hide();
      }
    };
    if (window.privilegeData && window.privilegeData.length > 0) {
      checkFoo(window.privilegeData);
    } else {
      // 将待判断的项目添加至数组，等待数据获取完毕后再进行判断
      window.privilegeAwaitItem ? window.privilegeAwaitItem.push({el: _el, binding: _binding}) :
      if (!window.privilegeLoading) {
        window.privilegeLoading = true;
        getPrivilegeData(({arr, obj}) => {
          // console.log(window.privilegeAwaitItem)
          window.privilegeData = arr;
          for (let item of window.privilegeAwaitItem) {
            checkFoo(window.privilegeData, item);
          }
          window.privilegeAwaitItem = null;
          window.privilegeLoading = false;
          // console.log(window.privilegeAwaitItem)
        });
      }
    }
  },
  update: function (el, binding) {
```

```
    update: function (el, binding) {
      // TODO
    },
    componentUpdated: function (el, binding) {
      // todo
    }
  })

  Vue.prototype.getPrivilegeData = getPrivilegeData;
  /**
   * 获取当前用户在当前页面内所拥有的所有权限
   * @param {Function} foo 回调函数，用于接收数据
   */
  function getPrivilegeData (foo) {
    console.log(window.activeMenu)
    let menuId = window.activeMenu || window.localStorage.getItem('activeMenu');
    Vue.prototype.axios.get('pass/v1/sysmenus/findFunctionMenusByUserId?menuType=2&companyId=' + w
      .then(res => {
          if (res.code === '0') {
              let arr = [],
                  obj = {};
              for (let item of res.data) {
                if (item.pId === menuId) {
                  arr.push(item);
                  obj[item.privilege] = true;
                }
              }
              foo({arr, obj});
          } else {
              foo({arr: [], obj: {}});
              this.$message.error(res.message)
          }
      }).catch(() => {
        foo({arr: [], obj: {}});
      });
  }
```

若现有业务在原有平台上改动过大，若必须更新建议与研发部联系，由研发部共同修改。