

User Research

背景与目标

将 Salesforce 的 User 相关模块（Profile / Role / Permission Set）的关键功能迁移到自研 NexusCRM。当前实现效果不佳，通过对 SF SOQL 数据和 NexusCRM 前端 JS bundle 的 deep research，定位到以下根本原因并给出调整方案。

Research 结论汇总

Salesforce 现状（pingcap.my.salesforce.com）

维度	数量/状态
Profiles	46 个（核心业务约 10 个）
Roles	43 个，5 层层级（Management → Region → Sub-region → Team → Individual）
Custom Permission Sets	139 个（业务相关约 15 个，其余为第三方工具集成）
Permission Set Groups	3 个
Active Standard Users	127 人
核心业务 Profile	Sales / Pre-Sales / Finance / Marketing / SDR / Management / Tech Support / System Administrator / Legal / DM

NexusCRM 现状（通过 JS bundle 逆向分析）

- **Tech Stack**: React 19 SPA + React Router v6 + TiDB Serverless + Vite
- **Auth**: JWT 存 `localStorage`（key: `auth_token`），通过 `/services/oauth2/userinfo` 端点获取用户上下文
- **JWT 现状**: 非 admin 用户的 token 仅含 `profile_id`（SF 原始 18 位 ID）+ `profile_name`，缺少 `role_id`、`permissions`
- **Setup 路由**: 存在完整路由定义，通过 `systemPermissions` 权限守卫控制访问，对 admin 用户可正常使用

已实现（代码层面）

功能	实现方式	状态
Setup 路由守卫	JB(perm) 检查 user.systemPermissions[perm]	✅ 逻辑正确
路由权限映射	/setup/profiles → ManageProfilesPermSets , /setup/roles → ManageRoles 等	✅ 完整
UserRoleId 字段	从 JWT role_id 读取存入 runtime context	✅ 代码已支持
对象权限 CRUD 检查	hasPermission(objectName, action) 基于 schema 的 retrieveable/createable/updateable/deletable	⚠ 实现了但数据源不对
系统权限定义	PermissionsManageUsers, PermissionsExportReport 等完整定义	✅ 已定义

未实现（代码层面）

功能	现状
Role 层级数据隔离	UserRoleId 存入 context 但查询 WHERE 从未引用它
Profile 级对象权限差异化	QB() 用 schema retrieveable/createable 给所有用户相同权限，Finance 和 Sales 看到完全相同的按钮
PermissionsViewAllRecords / PermissionsModifyAllRecords	代码中 硬编码为 false ，Permission Set 的 "View All" 授权无法生效
字段级权限	fieldPermissions 对象结构完整但数据为空数组，字段权限未从任何来源填充
Permission Set 执行	仅有记录存储，无前端判断逻辑（除 systemPermissions 路由守卫外）

核心问题诊断

问题 1：非 admin 用户 JWT 缺少 `role_id` 和 `permissions`（数据 + 后端问题）

表现：JWT payload 仅有 `profile_id` + `profile_name`，无 role 和 permissions 字段 根因：后端 `/services/oauth2/userinfo` 响应未为普通用户填充 `role_id` 和 `permissions` 影响：

- `UserId = undefined`：即使前端有 Role 隔离逻辑，也无法获取当前用户 role → 数据隔离完全失效
- `systemPermissions = {}`：Finance 用户无法访问 Setup（正确），但也无法正确获得本应有的导出等权限

问题 2：对象权限未按 Profile 差异化（代码问题，最直接影响用户体验）

表现：Finance 用户在 Opportunity 详情页仍能看到 Edit/Delete 按钮 **根因：**`QB()` 函数从 schema describe API 取 `retrieveable/createable/updateable/deletable`，这些字段反映数据库连接级别的权限，所有用户相同 **根因：**应该从 profile 的对象权限矩阵中读取，Finance 对 Opportunity 应为 ReadOnly **影响：**Profile 权限形同虚设，任何用户都能尝试编辑任何对象

问题 3：Role Hierarchy 数据隔离完全缺失（代码问题）

表现：Sales 用户 A 能看到 Sales 用户 B 的 Opportunity，Management 看到的数据与普通 Sales 相同 **根因：**`OT()` 查询构建器不添加任何 role 过滤条件；`UserId` 虽存入 context 但从未被查询引用 **影响：**数据隔离完全失效，所有用户看到同样的全量数据

问题 4：PermissionsViewAllRecords 硬编码 false（代码问题）

表现：Permission Set 分配了 "View All Accounts" 的用户实际上并不能查看所有 Account **根因：**`QB()` 中 `PermissionsViewAllRecords: !1`（即 `false`）为硬编码值，未从 Permission Set assignment 动态计算 **影响：**Permission Set 的 "View All" 类授权无法生效

问题 5：Report 权限管控方式与普通对象不同，当前完全未守卫（代码问题）

SF 中的 Report 权限机制（双层）：

层级	SF 实现	说明
系统权限	PermissionsRunReports（能否运行报表） + PermissionsExportReport（能否导出）	Profile 级系统权限，非 CRUD 矩阵
Folder 共享	FolderShare 记录控制哪些 User/Role/Group 能访问哪个报表文件夹	独立于 Profile 的二次过滤

Report 不走 Object CRUD 权限矩阵（`QB()` 机制对 Report 无意义）。

NexusCRM 现状（通过 JS bundle 确认）：

- `/analytics/report-builder` 路由**完全没有** `wd()` 权限守卫，任何已登录用户均可访问报表构建器
- `PermissionsRunReports` / `PermissionsExportReport` 在代码中**只有字段定义，从未被** `JB()` 调用检查
- Folder sharing API (`getFolderShares`, `addFolderShares` 等) 已实现，但仅用于 Setup UI 管理操作，**不作为运行时访问控制门**
- 实际效果：Finance 用户可以无限制地运行和导出任何报表

需要做的：

1. 对 `/analytics/*` 路由加 `wd({perm: "RunReports"})` 守卫（依赖 Phase 1 的 JWT permissions 补全）
2. 报表导出按钮增加 `JB("ExportReport")` 检查
3. 报表列表加载时按 Folder 共享关系过滤（当前 `FolderShare` API 已有，接入查询逻辑即可）

问题 6：User API 权限完全未检查（代码问题）

SF 中的 API 权限： `PermissionsApiEnabled` — Profile 系统权限，控制用户是否能够通过 API 访问 Salesforce（非 CRUD 矩阵）

NexusCRM 现状（通过 JS bundle 确认）：

- `PermissionsApiEnabled` 在代码中**只有字段定义**（在 `Jle` 对象）
- 整个 bundle 中 `JB()` 函数只被调用一次：`JB("ViewSetup")`
- API token / Open API 相关路由和功能**完全没有权限守卫**
- 实际效果：任何已登录用户都能生成和使用 API token，无法按 Profile 限制 API 访问

需要做的：

1. 在后端 `/services/oauth2/userinfo` 的 `permissions` 中返回 `ApiEnabled: true/false`（依赖 Phase 1）
2. API token 申请入口加 `JB("ApiEnabled")` 检查
3. 后端 API 中间层验证 token 归属用户的 `ApiEnabled` 权限

问题 7：Permission Set 迁移过多/未筛选（数据问题）

表现： SF 的 139 个 Permission Set 中大量是第三方工具集成（Chorus, CRUSH, PROLIFIQ, Zendesk, Zoom 等）
根因： 迁移时全量导入，未区分"业务逻辑 PSets"与"工具集成 PSets"
影响： CRM 中大量无用 Permission Sets 造成混乱，且在 CRM 中无法执行

问题 8： `profile_id` 使用 SF 原始 ID，权限矩阵未建立（数据问题）

表现：JWT 中 `profile_id: "00e5g000000yWvxAAE"` 是 SF 18 位记录 ID **根因：**CRM 的 `profiles` 表可能没有以此 ID 为主键的对象权限矩阵记录 **影响：**即使修复了 `QB()` 的读取逻辑，如果 `profiles` 表没有权限数据，仍然无法生效

调整方案

Phase 1: 补全 JWT payload (后端修复, 最基础)

目标：所有用户登录后 JWT/userinfo 响应包含完整权限上下文

调整后的 `/services/oauth2/userinfo` **响应结构：**

json

Code block

```
1  {
2    "user_id": "...",
3    "name": "...",
4    "email": "...",
5    "profile_id": "crm-internal-uuid",           ← 改为 CRM 内部 UUID (非 SF 18位 ID)
6    "profile_name": "Finance",
7    "role_id": "crm-internal-role-uuid",         ← 新增 (前端已有代码支持读取)
8    "role_name": "Finance",
9    "permissions": {                             ← 新增 (前端已有代码支持读取)
10     "ViewSetUp": false,
11     "ManageUsers": false,
12     "ManageProfilesPermSets": false,
13     "ManageRoles": false,
14     "ExportReport": true,
15     "ViewAllAccounts": false
16   }
17 }
```

后端工作：

- `users` 表确保有 `role_id` 外键 (指向 CRM 内部 roles 表)
- 登录时从 `users` → `roles` → `profiles` → `permission_set_assignments` 联查，组装 `permissions` 对象
- 将 SF `profile_id` → CRM 内部 UUID 的映射维护在 `profiles` 表中

Phase 2: 建立 Profile 对象权限矩阵并修复 `QB()` 读取逻辑 (数据 + 代码修复)

目标：不同 Profile 的用户在对象详情页看到不同的操作按钮

Step A — 数据：在 `profiles` 表建立权限矩阵

sql

Code block

```
1  -- profiles 表新增字段（或单独的 profile_object_permissions 表）
2  ALTER TABLE profiles ADD COLUMN object_permissions JSON;
3  -- 示例数据
4  -- Finance profile: Opportunity 只读
5  UPDATE profiles SET object_permissions = '{
6    "opportunity":
7      {"read":true,"create":false,"edit":false,"delete":false,"viewAll":false,"modify
8      All":false},
9    "account":
      {"read":true,"create":false,"edit":false,"delete":false,"viewAll":false,"modify
      All":false},
      "contact":
      {"read":true,"create":false,"edit":false,"delete":false,"viewAll":false,"modify
      All":false}
    }' WHERE name = 'Finance';
```

Profile	Account	Opportunity	Contact	Lead
System Administrator	CRUD+ViewAll	CRUD+ViewAll	CRUD+ViewAll	CRUD+ViewAll
Sales	CRUD	CRUD	CRUD	CRE
Pre-Sales	RE	R	RE	R
Finance	R	R	R	R
Marketing	RE	R	RE	CRUD
SDR	R	R	R	CRUD
Management	R	R	R	R
Tech Support	R	RE	RE	R
Legal	R	R	R	R
DM	R	R	R	R

Step B — 代码：修改 `QB()` 函数数据源

当前：从 schema `retrieveable/createable/updateable/deletable` 读取（对所有用户相同）

修改为：从后端 `/services/oauth2/userinfo` 响应中的 `object_permissions` 字段读取（per-profile）

Code block

```
1  userinfo response 新增:
2  "object_permissions": {
3    "opportunity":
4      {"read":true,"create":false,"edit":false,"delete":false,"viewAll":false,"modify
5      All":false},
6    ...
7  }
```

QB() 直接使用此数据而非 schema API 的 describe。

Phase 3: 实现 Role Hierarchy 数据隔离 (后端 + 数据修复, 核心难点)

目标: 用户只能看到自己 + 下级 Role 成员拥有的记录

Step A — 数据: 迁移 43 个 Role 到 CRM 的 `roles` 表

sql

Code block

```
1  CREATE TABLE IF NOT EXISTS roles (
2    id VARCHAR(36) PRIMARY KEY,
3    sf_role_id VARCHAR(20),           -- SF 原始 ID, 参考用
4    name VARCHAR(100),
5    parent_role_id VARCHAR(36),       -- 自引用, 构成层级
6    path VARCHAR(500)                -- "Management/APAC Region/APAC Sales
7                                     Team A"
8  );
```

层级结构 (已从 SF 提取) :

- L1 Root: Management
- L2: APAC Region / NA & EMEA Region / JP Region / Business Ops / Finance / Marketing / Legal
- L3: APAC Sales Team A/B/India / NA Region / EMEA Region / Japan Sales / Pre-Sales 各区域
- L4: 各 Team 下的 Leader / Member

Step B — 后端: NexusCRM 后端数据查询加入 Role 过滤

sql

Code block

```
1  -- 查询当前用户有权看到的 Opportunity (后端在 SQL 层注入)
```

```

2  SELECT o.* FROM opportunity o
3  JOIN users u ON o.owner_id = u.id
4  JOIN roles ur ON u.role_id = ur.id
5  WHERE ur.path LIKE CONCAT(
6    (SELECT path FROM roles WHERE id = :current_user_role_id), '%'
7  )

```

注意： `PermissionsModifyAllRecords = true` 的 Profile (System Admin) 和 `ViewAllRecords` Permission Set 用户绕过此过滤。

由于 NexusCRM 是浏览器直连 TiDB，数据隔离必须在服务器端中间层实现（不能依赖前端构建 WHERE），需确认是否有 backend proxy 层可以注入此条件。

Phase 4: 修复 PermissionsViewAllRecords (代码修复)

目标：拥有 "View All" 类 Permission Set 的用户能突破 Role 隔离看到全量数据

修改： `QB()` 函数中 `PermissionsViewAllRecords: !1` 改为从用户 `permissions` 字段动态读取：

js

Code block

```

1  // 修改前 (硬编码)
2  PermissionsViewAllRecords: false,
3  PermissionsModifyAllRecords: false
4
5  // 修改后 (从 userinfo 读取)
6  PermissionsViewAllRecords: userPermissions.viewAll?.[objectName] ?? false,
7  PermissionsModifyAllRecords: userPermissions.modifyAll?.[objectName] ?? false

```

对应后端需在 `permissions` 中返回：

json

Code block

```

1  "view_all_objects": ["account", "opportunity"],
2  "modify_all_objects": []

```

Phase 5: 精简 Permission Set 数据 (数据修复)

删除（工具集成类，CRM 中无对应功能）：

- CRUSH / PROLIFIQ / Elvance（第三方 Account Plan）
- Chorus / Zendesk / ZoomInfo / LinkedIn（第三方集成）
- C2C / CASC / Einstein / Highspot / Koncert（SF 原生 / 销售工具）

保留并在 CRM 中实现的核心 PSets（约 15 个）：

Permission Set	CRM 对应功能
Export Report	报表导出按钮权限
View All Accounts	绕过 Role 隔离查看所有 Account
View All Opportunities	绕过 Role 隔离查看所有 Opp
Edit All Leads	突破 Role 隔离编辑所有 Lead
Forecast	Forecast 模块访问
Transfer Record Access	改变记录 OwnerId
Open API Access	API token 功能
Sales User	Sales 扩展权限
Content access	文档/附件访问

Phase 6：修复 Report 和 API 权限守卫（代码修复）

⚠ 注意：Report 权限在 SF 中不走 CRUD 矩阵，是独立的系统权限 + Folder 共享两层机制，CRM 需单独处理。

目标 A — Report 访问控制：

1. 系统权限守卫（依赖 Phase 1 JWT 补全）：

- `/analytics/*`、`/dashboards/*` 路由加 `wd({perm: "RunReports"})` 守卫
 - 报表导出按钮加 `JB("ExportReport")` 检查
 - 对应后端 `permissions` 返回：`RunReports: true/false`、`ExportReport: true/false`
2. Folder 共享过滤（report 列表运行时过滤）：
- 报表列表加载时调用已有 `getFolderShares(folderId)` API，过滤当前用户无权访问的 Folder
 - Folder 共享权限来源：`FolderShare` 表中 `SharedTo` = 当前用户的 `UserId / RoleId / GroupId`
 - ⚠ 这是 **独立于 Profile 权限矩阵** 的第二层过滤，不应并入 `QB()` 机制

Profile 对 Report 的权限对照（SF 标准）：

Profile	RunReports	ExportReport	备注
System Administrator	✓	✓	且能访问所有 Folder
Sales / Pre-Sales / SDR	✓	✓	只能访问被共享的 Folder
Finance / Marketing	✓	✓	只能访问被共享的 Folder
Management	✓	✓	通常共享所有业务 Folder
Tech Support / Legal / DM	✓	✗	可查看不可导出

目标 B — API 访问控制：

- 1. 后端 /services/oauth2/userinfo permissions 新增 ApiEnabled: true/false
- 2. API token 申请入口加 JB("ApiEnabled") 检查
- 3. 后端 API 中间层：验证请求 token 对应用户的 ApiEnabled 权限（防止绕过前端）

Profile 对 API 的权限对照：

Profile	ApiEnabled	备注
System Administrator	✓	-
Sales / Pre-Sales	✓	需要访问 CRM 数据同步
Finance / Marketing / SDR	✗	只需 UI 操作，无 API 需求
Management	✓	报表/数据导出需要

执行优先级

Code block

- 1 Phase 1（2-3天）：后端补全 JWT/userinfo 的 role_id + permissions → 所有权限检查的基础
- 2 Phase 2（3-5天）：Profile 对象权限矩阵建数据 + 修复 QB() 读取源 → 按钮级权限生效
- 3 Phase 3（5-10天）：Role Hierarchy 数据 + 后端 SQL 过滤 → 记录级隔离（最复杂）

- 4 Phase 4 (1-2天) : 修复 PermissionsViewAllRecords 硬编码 → Permission Set View All 生效
- 5 Phase 5 (1-2天) : 清理无用 PSets, 只保留核心业务 PSets
- 6 Phase 6 (2-3天) : Report + API 权限守卫 → 报表按 Folder 隔离 + API 访问按 Profile 控制

验证方法

1. **JWT 补全**: Finance 用户登录后, 解码 JWT 可见 `role_id`、`permissions.ExportReport: true`、`permissions.ApiEnabled: false` 等字段
2. **对象权限**: Finance 用户访问 Opportunity 详情页 → 无 Edit/Delete 按钮; Sales 用户有
3. **Role 隔离**: Sales (APAC Team A) 登录 → Opportunity 列表只含 APAC Team A 及其下级成员的记录; Management 登录看全量
4. **View All PSets**: 分配了 "View All Accounts" PSets 的 Finance 用户能看到所有 Account 记录 (突破 Role 隔离)
5. **Setup 访问**: admin@test.com 可访问 `/setup/profiles`、`/setup/roles`、`/setup/users` (已验证可用)
6. **Report 权限**: Tech Support 用户访问 `/analytics/report-builder` → 被拦截重定向 (`RunReports: false`); Finance 用户可访问报表列表但只看到被共享的 Folder 下的报表; Finance 用户报表页无"导出"按钮 (`ExportReport: false`)
7. **API 权限**: Finance 用户访问 API token 页面 → 无申请入口 (`ApiEnabled: false`); Sales 用户可见并申请

关键文件

- **NexusCRM 前端 bundle**: `/assets/index-CjCSuFqm.js` (本地缓存 `/tmp/nb.js`)
 - `QB()` — 对象权限构建, 需修改数据源 (约在 bundle 中 `PermissionsDelete` 附近)
 - `JB(perm)` — 系统权限检查 hook, 检查 `systemPermissions[perm]`
 - `OT()` — 查询构建器, 无 role 过滤, 需在后端层面拦截
 - `nD mapping` — `/setup/*` 路由 → 所需权限映射表
- **TiDB 表**: `users` / `profiles` / `roles` / `permission_sets` / `permission_set_assignments`
- **后端 endpoint**: `/services/oauth2/userinfo` (需补全 `role_id` + `permissions` + `object_permissions`)

- **GitHub Repo:** `pingcap-inc/MetaDrivenPlatform4AI` (私有)
- **SF 配置参考:** 46 profiles / 43 roles / 139 PSets (已通过 SOQL 全量提取)