数据库配置可用性测试分析

作者：朱菁菁

时间：20190719

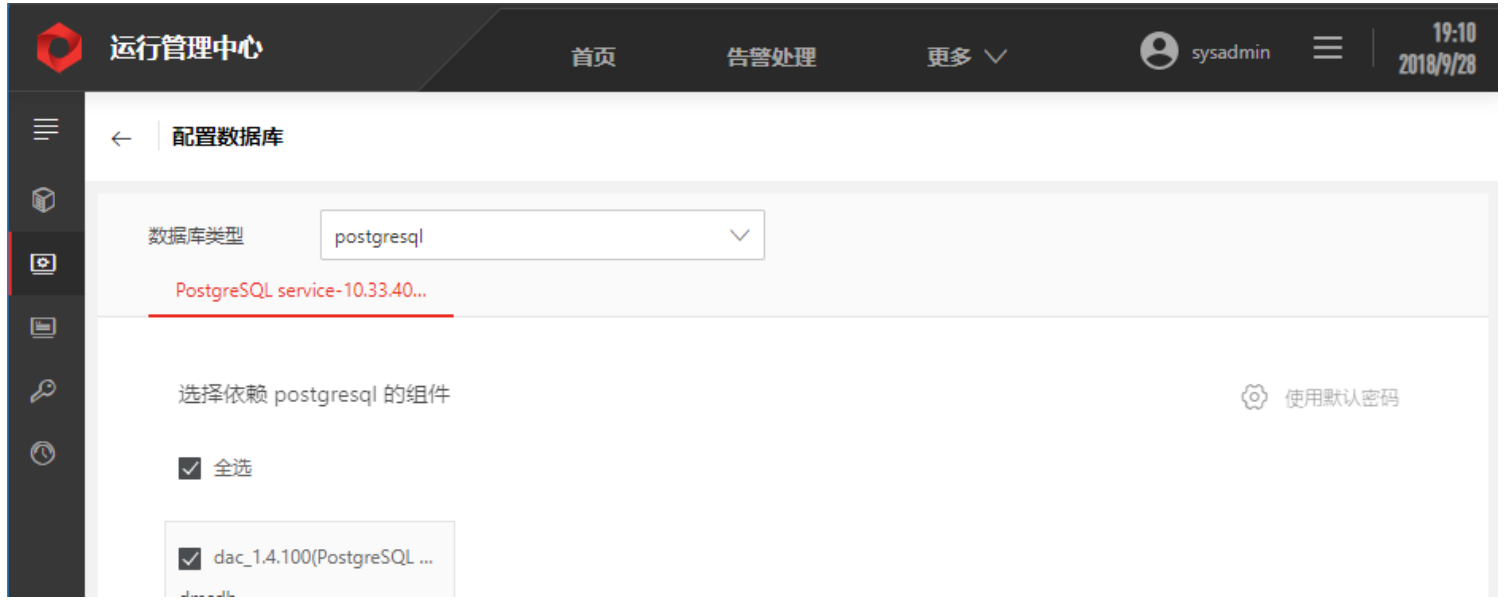
目录

1 背景及目的

阐述本文背景目的及场景分析

背景

运行管理中心v1.2.0产品在项目现场进行部署的时候，从现场调研中我们收到了关于部署环节中用户不知道如何配置数据库的问题反馈，由于不清楚如何配置数据库，导致安装部署流程受阻，对项目现场来说不利于操作，降低了工作效率，该问题亟需优化和解决。通过收集用户问题，了解用户使用背景和困惑后，初步设计了优化方案，为了在前期保证方案可行性，需要通过可用性测试分析，帮助评估设计方案，明确设计优化点合理性与可行性，进而推进落地。



（优化前界面）

目的

- 分析了解用户配置数据库的操作流程和需求。
- 通过可用性测试确定设计优化方向与设计方案。

背景及目的

目标用户

分公司技术支持（几乎常年奔赴在一线，为项目进行部署配置。）

场景分析

多台服务器需要安装平台产品时，Linux版产品下，技术支持安装完运管中心后，需要通过中心安装其他业务组件；Windows版产品下，技术支持安装完运管中心后和中心服务器上的组件后，需要通过运管中心安装其余服务器上的组件。
此时技术支持进入软件安装页面，进行多机批量安装，此时若有多个组件依赖数据库，则需要技术支持选择数据库、选择数据库实例、选择依赖这个数据库实例的组件，若客户有要求，还需要根据客户要求配置数据库的连接用户名和密码。

这里涉及到需要用户操作的对象有：数据库、数据库实例、组件、用户名、密码，动作包括选择、修改、确定，设计上需要理清楚对象与操作之间的关系，做到布局合理，操作清晰。同时，用户对于数据库的配置，在线下有相应的配单，且已经形成一些惯有配置的流程，因此，最好是贴合用户使用习惯进行优化设计，减少用户的学习成本。

2

方案分析与设计

说明分析过程以及阐述设计方案

方案分析与设计 – 用户画像

为了更好的了解用户和使用场景，前期特意针对参与项目部署的用户做了调研，并建立了用户画像以帮助了解用户和场景，帮助分析问题。

通过用户画像，我们可以发现用户在安装环节配置数据库的阶段，比较关注安装便捷性，希望步骤清晰，操作简单，在做配置这类业务性较强的操作时，期望获得良好的引导。

高建　分公司技术支持

基本信息

- 24岁，大学本科，计算机专业
- 1年工作经验，每天工作12个小时以上
- 经常在项目现场

目标

- 快速部署交付
- 准确识别、排查、解决现场的问题
- 提高工作效率，减少问题重复发生
- 让客户满意

职责和特征

- 快速高效完成项目部署，在项目部署完后给客户简单培训
- 及时解决现场产品问题，保证问题解决效率
- 向研发反馈现场的问题需求

关注点

- 前期关注现场网络、硬件、软件环境情况，关心产品能否适应
- 了解客户所需服务和功能，相应组件和模块，关心服务器规划和部署方案的合理有效
- 关注安装便捷性，希望可以在安装配置过程中简化操作
- 需要了解业务配置、功能调试的完整过程和实现效果

痛点

- 部署方案不合理不详细的情况较多
- 安装说明不清楚，导致安装过程出现问题，如缺少对平台或客户端的依赖数据库的说明
- 安装步骤较繁琐，数据库、脚本、server都是单独安装
- 业务配置需要在多个模块之间跳转，缺少引导说明，如果不仔细看文档的话较容易配错

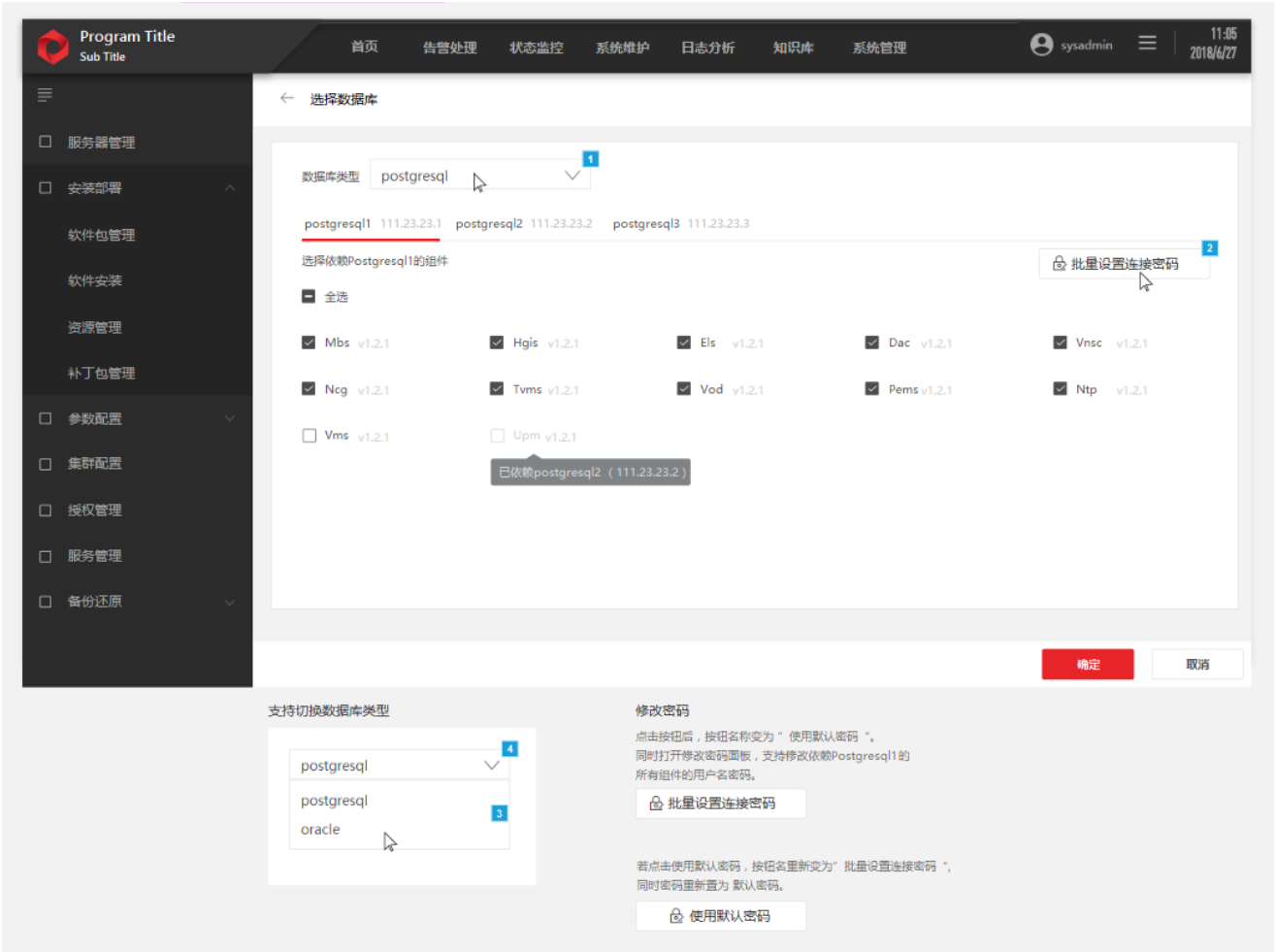
（图片来源：网络）

“和售前、销售、总部的沟通很重要，决定部署方案的质量。”

原方案问题

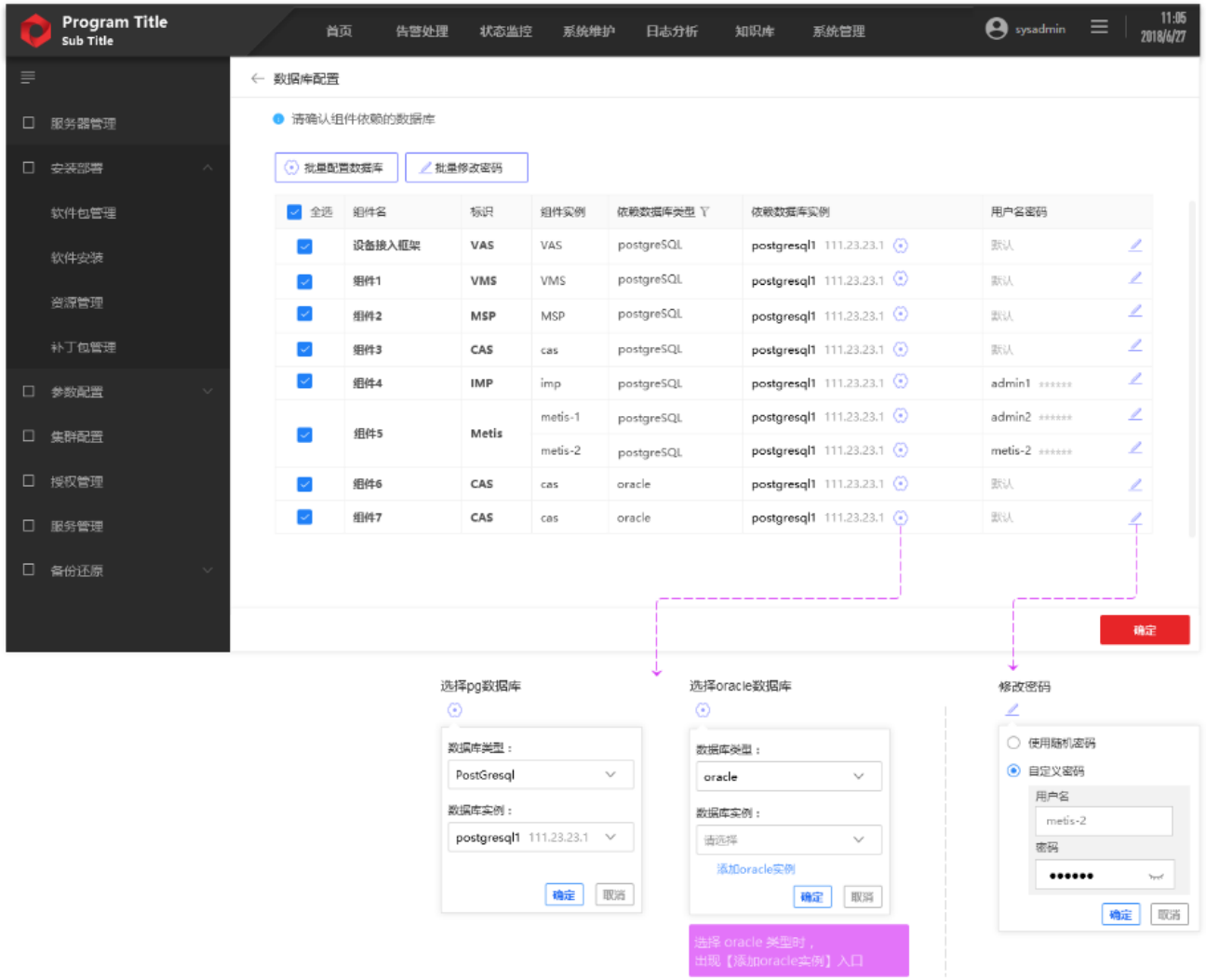
在新方案设计之前，先对用户进行了访谈，了解用户的问题、困惑和感受。

- 用户原本的预期是系统已经帮忙设置好了，用户只需要走一下配置数据库的流程，直接确认即可下一步操作；
- 用户面对当前的配置页，不知道勾选了某个组件后还需要怎么操作；
- 用户从布局上不能对所有数据库有清楚的了解；



分析：

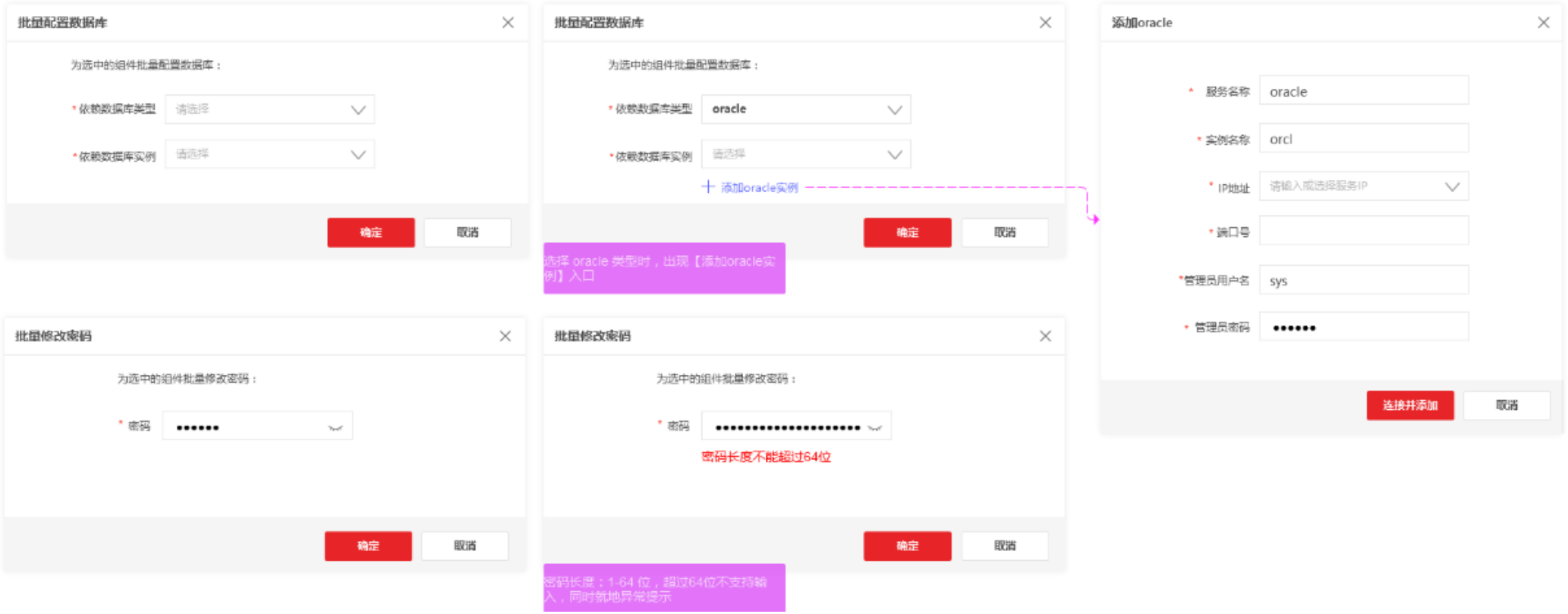
- 布局不清晰，无法直观确认所有数据库配置情况；
- 在项目中数据库类型较少、数量不多的场景下，隐藏较深，切换查看较麻烦；
- 功能上未提供默认配置，操作麻烦；
- 缺少有效操作引导，导致用户迷茫，操作受阻。



该方案改变原有布局，调整为列表结构。
主要以组件维度展示数据库配置情况，采用列表方案进行页面信息呈现。

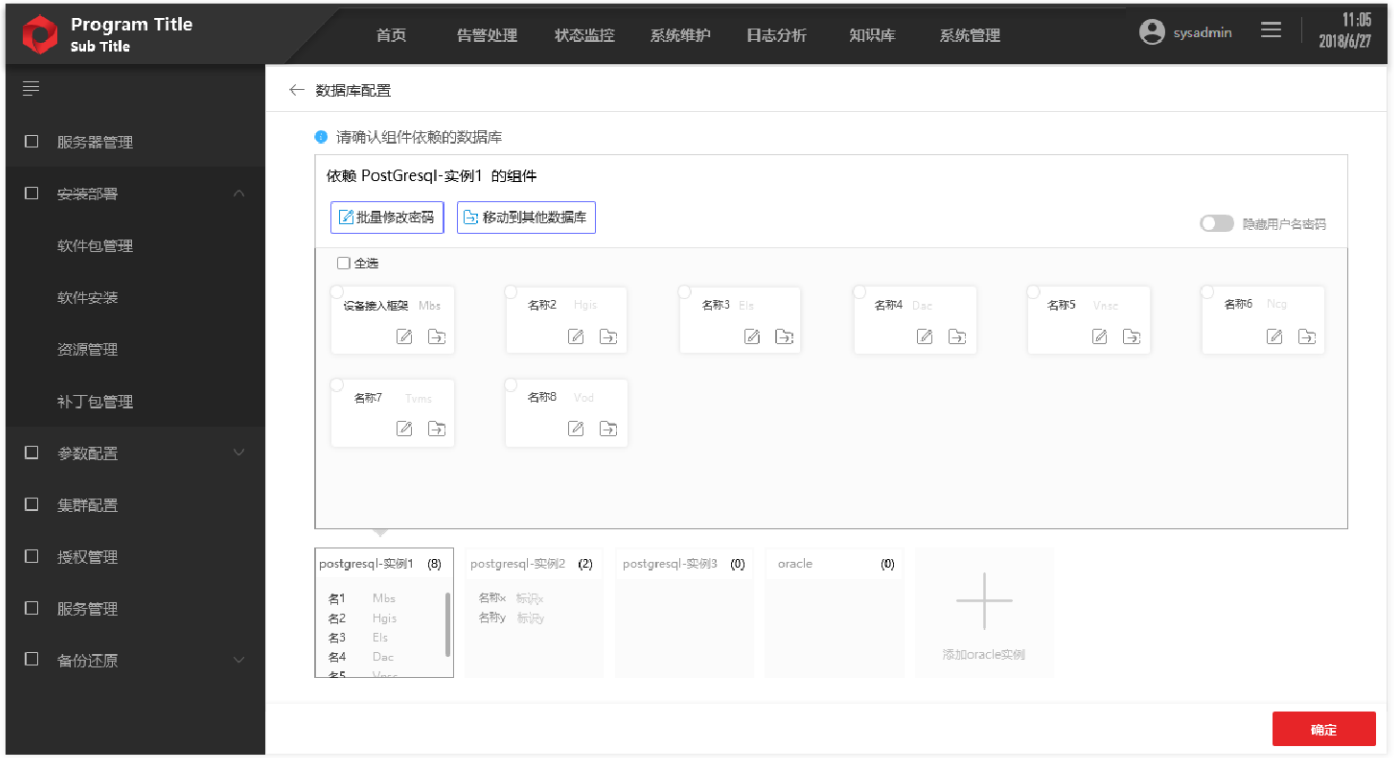
- 将标题改为“数据库配置”，表明该页是进行数据库的配置；
- 进入模块页的时候提供默认配置关系，支持修改；
- 顶部提示语用于引导告知用户执行的是“确认组件依赖的数据库”的任务，只需要用户进行配置信息的确认；
- 通过表格从组件视角进行数据库的选择及密码设置；
- 期望通过表格和默认值使用户查看到组件与所依赖数据库的对应关系，并且如果组件与默认数据库没有问题，那么只需要用户直接确认，无需进行配置操作。如果多个组件需要配置其他数据库，也支持批量修改。

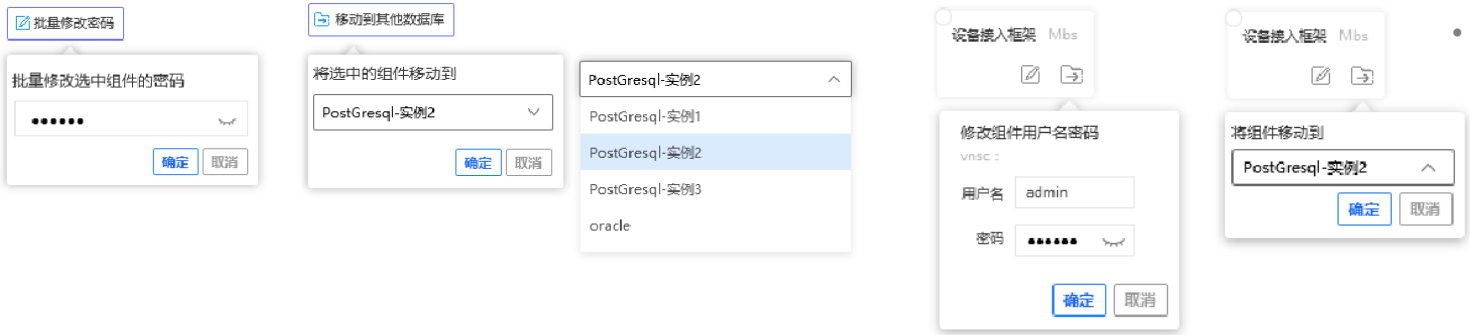
支持用户进行批量操作：



- 优点：直观展示所有组件，以及组件对应配置的数据库信息。
- 缺点：页面改动太大，对习惯原有查看方式的用户来说学习成本提高 。

方案分析与设计 – 方案二



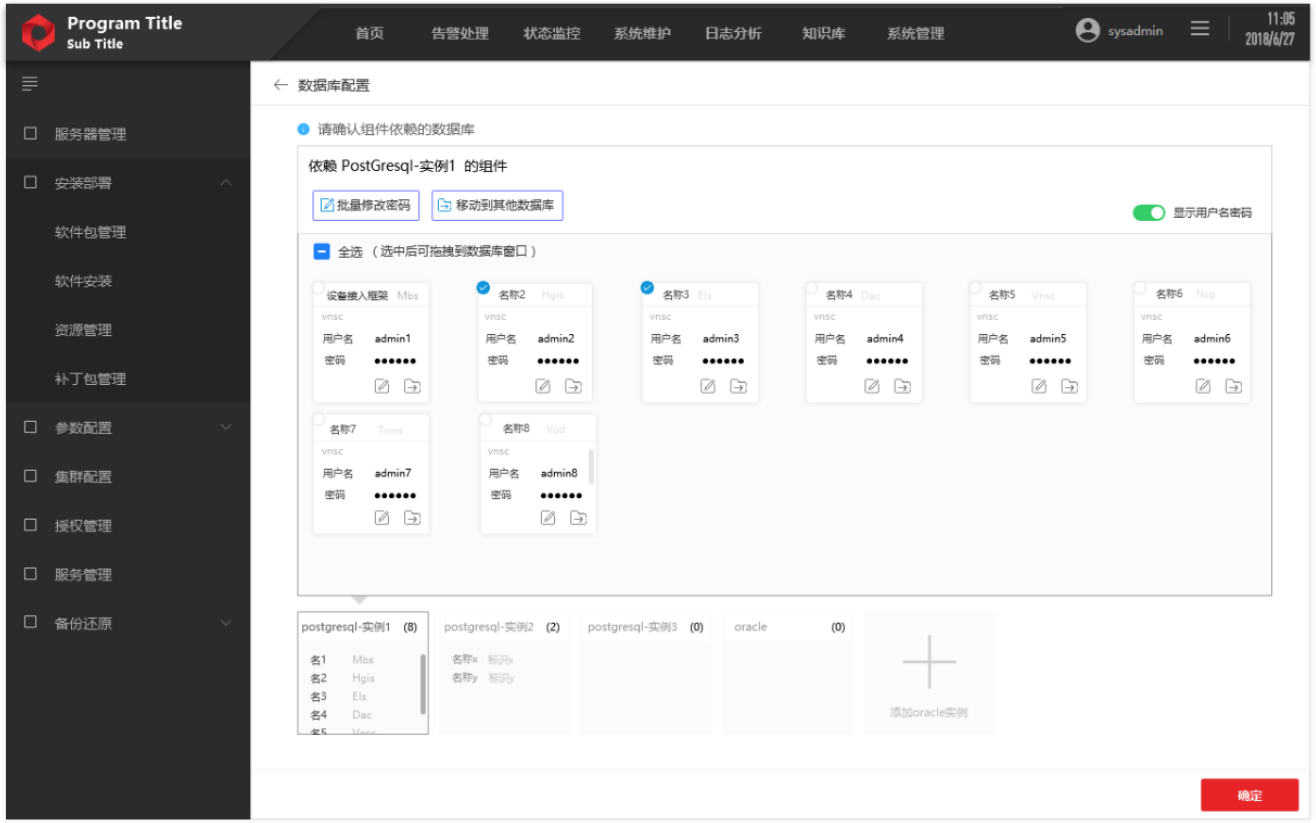


由于这里主要是为组件进行对应数据库的配置，最终目标是将所有需要配置数据库的组件根据提供的配单合理配置已装的数据库，因此该方案引入了组件公共池的概念，且将每个数据库进行平铺，形似“数据库篮子”。

- 将所有需要进行数据库配置的组件展示在一个组件公共池中，可以看到组件是否已被选择（配置）的情况；
- 将数据库设计成缩略窗口，可以全局看到每个数据库中已选择的组件有哪些，有多少；
- 支持切换数据库，从公共池窗口中看到组件勾选的情况并支持取消选择组件；
- 支持在界面中直接添加oracle数据库；
- 对于每个组件卡片，支持单独修改密码、移动到其他数据库，也支持批量修改和移动。

方案分析与设计 – 方案二

选择显示用户名密码时，支持查看/修改密码



拖拽操作示意：

这里还初步考虑了拖拽的交互方式，期望通过这种模式支持用户将组件直接“扔”到他想配置的数据库窗口中。

- 优点：公共池概念贴合用户场景的需求，数据库缩略图的展示帮助用户直观了解配置关系和数量，不需要切换才能查看，操作更便捷。
- 缺点：拖拽模式前期学习成本较高

方案分析与设计 – 方案三

- 把所有已有的数据库展示出来；
- 若数据库窗口太多，超过页面宽度时，增加左右箭头的功能图标；

- 若所有组件都只支持pg实例，那么展示的oracle实例窗口及添加入口置灰，不支持点击，移入给出原因提示。

第三个方案是在可用性测试期间，从用户反馈中发现方案二评价较符合用户心理模型，所以在方案二的基础上做调整得出的，依旧沿用了“组件公共池”概念。

- 布局上优先展示数据库，后组件；
- 操作逻辑上，引导先选数据库，再选组件，最后确认和保存数据库的配置；
- 所有组件的已添加情况在底部操作旁边有文案提示；
- 未将所有组件添加完毕，则按钮置灰，不操作；
- 默认值上，若组件只支持pg，且系统内有pg实例，则初始页中该组件默认选中添加到第一个pg实例，oracle相同情况逻辑保持一致；
- 如果系统内无oracle实例，则组件不被默认勾选添加，支持用户直接配置oracle库；
- 支持切换选择密码方式，使用系统随机密码或者用户自定义密码；

3 可用性测试结果分析

王沅松

公安技术支持

用户数据库配置经验：

1.大部分场景只有一个数据库，很少的情况下，现场需要装几个数据库，可能是以此来提高性能。

2.有些组件不支持某一种数据库，所以也可能要装几个库。

用户评价：

方案	星级	排名	结论	分析
方案一	★	最差	【感受】一看到觉得不直观，不好看 【思维】1.有一个以上PG数据库的时候，看不出来差别； 2.不知道怎么切数据库	列表样式不直观； 引导性差； 列表数据区分度很低；
方案二	★ ★ ★	第一	【感受】 1.一开始看到觉得有点乱->先看到组件，没注意到下面的数据库 2.注意到“移动到”按钮，但是对移动到哪里感到疑惑->仔细看了会儿后知道怎么移动组件”，是先选中-点移动到-选择数据库实例-点击数据库实例查看“	结构排布不符合用户行为习惯； 数据库窗口不突出，应该提高数据库的凸显； 不常见的操作词汇使用户产生疑惑，操作用词应贴合用户操作场景；

韩飞

公安技术支持

用户数据库配置经验：

没有直接配过数据库，认为一般情况下一个库应该就够了。

用户在配置数据库上认为在运管里应该将一些默认安装好的数据库和后来加上的区分开。

用户评价：

方案	星级	排名	结论	分析
方案一	★	最差	【感受】没原来交互方案好，不直观 【产品理解和操作】一进来看到一大堆组件，我要做的事情是将组件关联到数据库上，这个页面里看不清楚数据库，不直观	列表展示方式反而导致关联关系无法直观查看； 满满的表格数据给用户心理造成压迫感；
方案二	★ ★ ★	第二	【感受】看起来很清楚，能够看明白先做什么后做什么，将操作、组件、数据分析区分开了。 数据库配置应该不是频繁操作的配置，但这种设计看起来让人觉得很容易去改变、移动。	数据库、组件、操作需要有明显的区分； 提供默认配置项，弱化除选择配置外的其他操作，避免引起用户误操作；
方案三	★ ★ ★ ★ ★	第一	【感受】直观，将数据库和公共组件库两个概念，以及相应的数据区分开。	数据库、组件、操作区分明显，有助于用户确认和操作。

朱正清

技术支持

用户数据库配置经验：

多次配置数据库有较为丰富的经验，对业务和场景比较熟悉。

用户评价：

方案	星级	排名	结论	分析
方案二	★	最差	【感受】一开始没注意到数据库（在页面下面看不到） 【产品理解和操作】 认为应该先选数据库->选组件->点确定，和本方案略有不同； “移动到”什么意思，和迁移数据库一样么？对移动到的对象不理解	数据库窗口不突出，布局上优先数据库，后组件； 操作逻辑：先选数据库，再选组件，最后确认和保存； 不常见的操作词汇使用户产生疑惑；
方案三	★ ★ ★ ★ ★	第一	【感受】可以很清楚的知道哪些组件在什么库下 【产品理解和操作】 1.将数据库和公共组件库区分开了 2.对于选择组件的方式，认为像方案二的移动不合适，一般操作者不知道我们的产品可以移动，只会点击。倾向于勾选的方式。	数据库、组件、操作需要有明显的区分； 用户操作倾向于使用频率最高的勾选，容易使用户理解。

通过对以上三种方案的可用性测试和分析，获得以下结论：

分析结论	理由
1.数据库、组件、操作需要有明显的区分	用户需求是能够清楚的看到并区分组件、数据库、操作
2.布局上优先数据库，后组件	用户在看到组件的时候，也要直观看到数据库，方便建立联系。 组件数量多，区域面积大，视觉很突出，容易关注；数据库个数少，如果所占面积小，会造成视觉不明显，所以放前置位突出数据库，方便用户关注并按照业务建立连接关系。
3.操作逻辑：先选数据库，再选组件，最后确认和保存该数据库的配置，设计公共组件池的功能和概念	对于项目来说，用户对数据库配置的操作是低频操作，需要依据用户配置习惯，减少用户认知冲突
4.弱化操作，对于组件仅提供选择操作，减少其他复杂操作	用户希望在一个页面里看到所有组件，方便对组件进行分配、移动

4

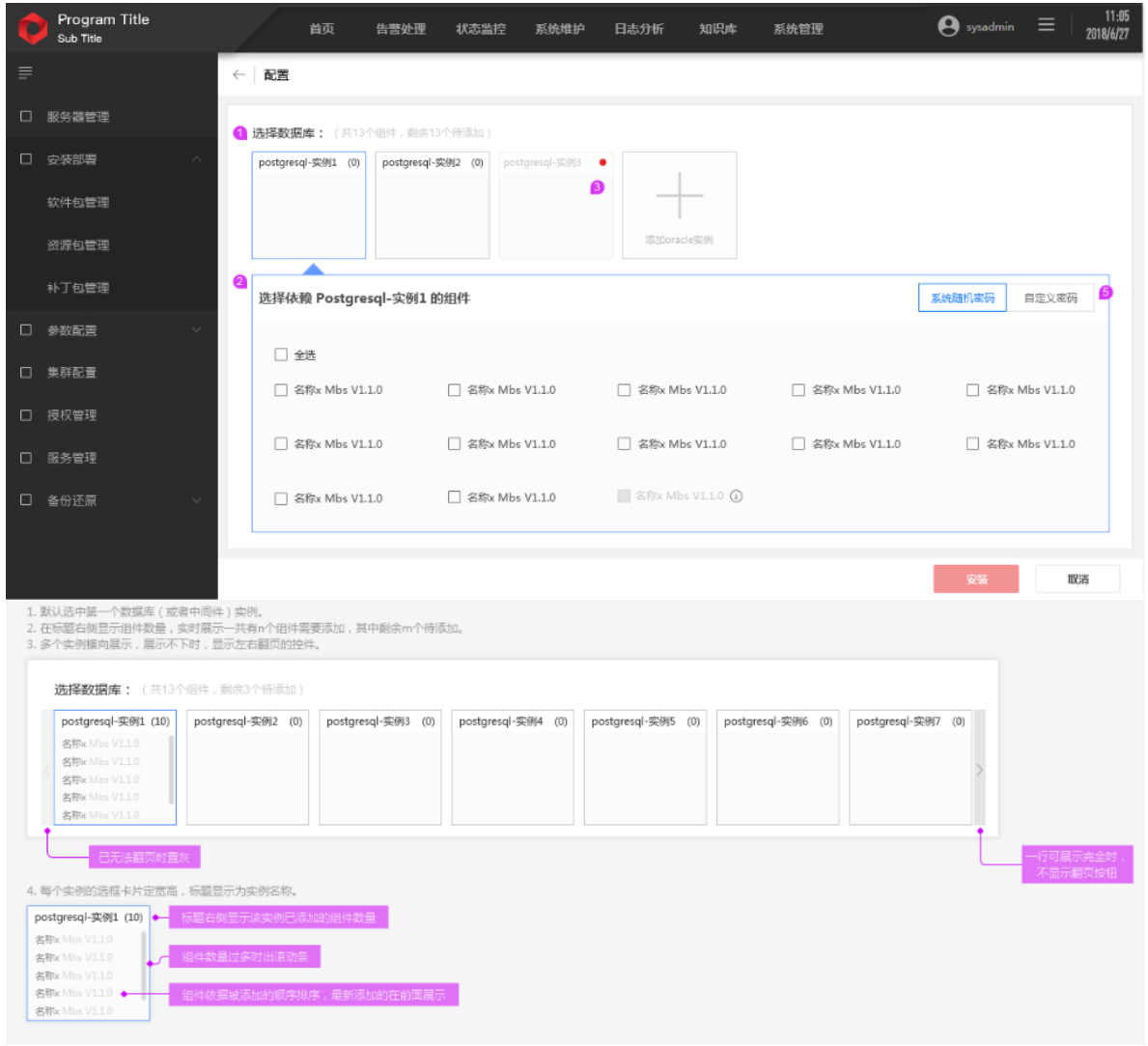
总结

总结及最终方案

总结

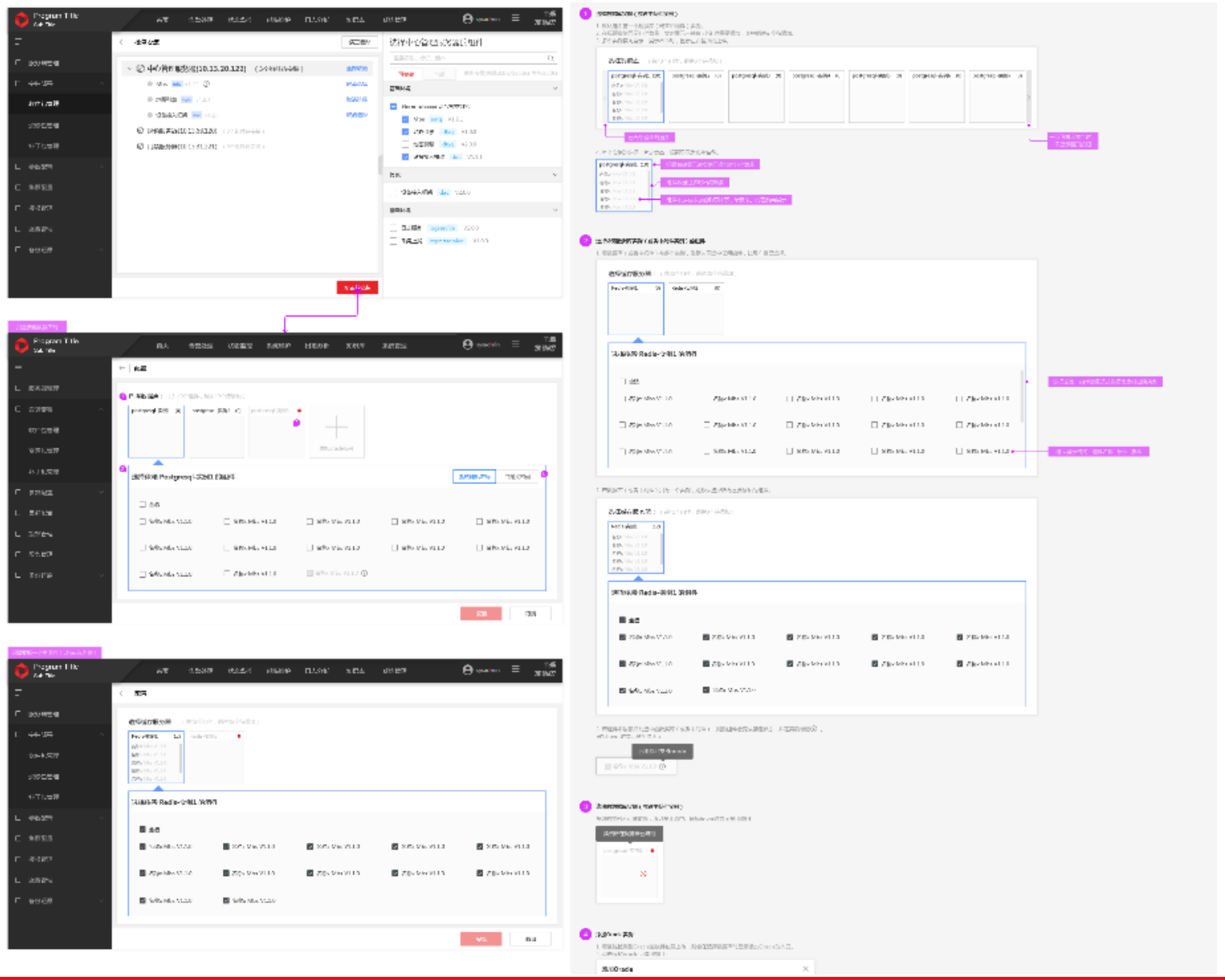
经过可用性测试分析，可以发现用户需要能够明显区分数据库和组件，布局上优先突出数据库，操作逻辑上需要先选择数据库，再选择组件，最后确定信息。在操作上，要根据用户行为习惯弱化其他不必要的操作，使界面便于操作和查看核对。

最后在方案三基础上进一步优化输出最终方案。



总结 – 最终方案

最终方案通过用户验证，并已落地。



THANKS