SAAS

# 云服务的三种模式

IaaS（Infrastructure as a Service基础设施即服务）是指把IT基础设施作为一种服务通过网络对外提供。在这种服务模型中，用户不用自己构建一个数据中心，而是通过租用的方式来使用基础设施服务，包括服务器、存储和网络等。在使用模式上，IaaS与传统的主机托管有相似之处，但是在服务的灵活性、扩展性和成本等方面IaaS具有很强的优势

PaaS是（Platform as a Service）的缩写，是指[平台即服务](https://baike.baidu.com/item/%E5%B9%B3%E5%8F%B0%E5%8D%B3%E6%9C%8D%E5%8A%A1/4329761" \t "_blank)。 把[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8/100571" \t "_blank)平台作为一种服务提供的商业模式，通过网络进行程序提供的服务称之为SaaS(Software as a Service)，而[云计算](https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97" \t "_blank)时代相应的服务器平台或者开发环境作为服务进行提供就成为了PaaS(Platform as a Service)

SaaS即Software-as-a-Service（软件即服务）是随着互联网技术的发展和应用软件的成熟， 在21世纪开始兴起的一种完全创新的软件应用模式。传统模式下，厂商通过License将软件产品部署到企业内部多个客户终端实现交付。SaaS定义了一种新的交付方式，也使得软件进一步回归服务本质。企业部署信息化软件的本质是为了自身的运营管理服务，软件的表象是一种业务流程的信息化，本质还是第一种服务模式，SaaS改变了传统软件服务的提供方式，减少本地部署所需的大量前期投入，进一步突出信息化软件的服务属性，或成为未来信息化软件市场的主流交付模式。

企业开发

服务器(cpu、存储、网络) 应用

云服务

IAAS PAAS SAAS

基础设施即服务 平台即服务 应用即服务

# 多租户saas平台的数据库方案

一、对多租户的理解

多租户定义：多租户技术或称多重租赁技术，简称SaaS，是一种软件架构技术，是实现如何在多用户环境下（此处的多用户一般是面向企业用户）共用相同的系统或程序组件，并且可确保各用户间数据的隔离性。简单讲：在一台服务器上运行单个应用实例，它为多个租户（客户）提供服务。从定义中我们可以理解：多租户是一种架构，目的是为了让多用户环境下使用同一套程序，且保证用户间数据隔离。那么重点就很浅显易懂了，多租户的重点就是同一套程序下实现多用户数据的隔离。对于实现方式，我们下面会讨论到。

在了解详细一点：在一个多租户的结构下，应用都是运行在同样的或者是一组服务器下，这种结构被称为“单实例”架构（Single Instance），单实例多租户。多个租户的数据是保存在相同位置，依靠对数据库分区来实现隔离操作。既然用户都在运行相同的应用实例，服务运行在服务供应商的服务器上，用户无法去进行定制化的操作，所以这对于对该产品有特殊需要定制化的客户就无法适用，所以多租户适合通用类需求的客户。那么缺点来了，多租户下无法实现用户的定制化操作。

在翻阅多租户的资料时，还有一个名词与之相对应，那就是单租户SaaS架构（也被称作多实例架构（Multiple Instance））。单租户架构与多租户的区别在于，单租户是为每个客户单独创建各自的软件应用和支撑环境。单租户SaaS被广泛引用在客户需要支持定制化的应用场合，而这种定制或者是因为地域，抑或是他们需要更高的安全控制。通过单租户的模式，每个客户都有一份分别放在独立的服务器上的数据库和操作系统，或者使用强的安全措施进行隔离的虚拟网络环境中。因为本篇主要是讨论多租户，所以单租户的相关知识就简单了解一下，不做过多的阐述了。

二、多租户数据隔离的三种方案

在当下云计算时代，多租户技术在共用的数据中心以单一系统架构与服务提供多数客户端相同甚至可定制化的服务，并且仍可以保障客户的数据隔离。目前各种各样的云计算服务就是这类技术范畴，例如阿里云数据库服务（RDS）、阿里云服务器等等。

多租户在数据存储上存在三种主要的方案，分别是：

1. 独立数据库

这是第一种方案，即一个租户一个数据库，这种方案的用户数据隔离级别最高，安全性最好，但成本较高。

优点：

为不同的租户提供独立的数据库，有助于简化数据模型的扩展设计，满足不同租户的独特需求；如果出现故障，恢复数据比较简单。

缺点：

增多了数据库的安装数量，随之带来维护成本和购置成本的增加。

这种方案与传统的一个客户、一套数据、一套部署类似，差别只在于软件统一部署在运营商那里。如果面对的是银行、医院等需要非常高数据隔离级别的租户，可以选择这种模式，提高租用的定价。如果定价较低，产品走低价路线，这种方案一般对运营商来说是无法承受的。

2. 共享数据库，独立 Schema

这是第二种方案，即多个或所有租户共享Database，但是每个租户一个Schema（也可叫做一个user）。底层库比如是：DB2、ORACLE等，一个数据库下可以有多个SCHEMA

优点：

为安全性要求较高的租户提供了一定程度的逻辑数据隔离，并不是完全隔离；每个数据库可支持更多的租户数量。

缺点：

如果出现故障，数据恢复比较困难，因为恢复数据库将牵涉到其他租户的数据；

如果需要跨租户统计数据，存在一定困难。

3. 共享数据库，共享 Schema，共享数据表

这是第三种方案，即租户共享同一个Database、同一个Schema，但在表中增加TenantID多租户的数据字段。这是共享程度最高、隔离级别最低的模式。

即每插入一条数据时都需要有一个客户的标识。这样才能在同一张表中区分出不同客户的数据。

优点：

三种方案比较，第三种方案的维护和购置成本最低，允许每个数据库支持的租户数量最多。

缺点：

隔离级别最低，安全性最低，需要在设计开发时加大对安全的开发量； 数据备份和恢复最困难，需要逐表逐条备份和还原。

如果希望以最少的服务器为最多的租户提供服务，并且租户接受牺牲隔离级别换取降低成本，这种方案最适合。

在SaaS实施过程中，有一个显著的考量点，就是如何对应用数据进行设计，以支持多租户，而这种设计的思路，是要在数据的共享、安全隔离和性能间取得平衡。

三、选择合理的实现模式

衡量三种模式主要考虑的因素是隔离还是共享。

成本角度因素

隔离性越好，设计和实现的难度和成本越高，初始成本越高。共享性越好，同一运营成本下支持的用户越多，运营成本越低。

安全因素

要考虑业务和客户的安全方面的要求。安全性要求越高，越要倾向于隔离。

从租户数量上考虑

主要考虑下面一些因素

系统要支持多少租户？上百？上千还是上万？可能的租户越多，越倾向于共享。

平均每个租户要存储数据需要的空间大小。存贮的数据越多，越倾向于隔离。

每个租户的同时访问系统的最终用户数量。需要支持的越多，越倾向于隔离。

是否想针对每一租户提供附加的服务，例如数据的备份和恢复等。这方面的需求越多， 越倾向于隔离

技术储备

共享性越高，对技术的要求越高。

四、建议和意见

试用版和正式版

试用版由于是免费的使用共享数据库和共享数据表

正式版及用户花钱购买了我们就可以选择第二种甚至是第一种

Schema和DataBase是否等同？

涉及到数据库的模式有很多疑惑，问题经常出现在模式和数据库之间是否有区别，如果有，区别在哪里。

**取决于数据库供应商**

对schema（模式）产生疑惑的一部分原因是数据库系统倾向于以自己的方式处理模式

（1）MySQL的文档中指出，在物理上，模式与数据库是同义的，所以，模式和数据库是一回事。

（2）但是，Oracle的文档却指出，某些对象可以存储在数据库中，但不能存储在schema中。 因此，模式和数据库不是一回事。

（3）而根据这篇SQL Server技术文章[SQLServer technical article](https://technet.microsoft.com/en-us/library/dd283095(v=sql.100).aspx)，schema是数据库SQL Server内部的一个独立的实体。 所以，他们也不是一回事。

因此，取决于您使用的RDBMS，模式和数据库可能不一样。

创建Schema

尽管上述三个DBMS在定义schema方面有所不同，还是有一个共同点，就是每一个都支持CREATE SCHEMA语句。

MySQL

在MySQL中，CREATE SCHEMA创建了一个数据库，这是因为CREATE SCHEMA是CREATE DATABASE的同义词。 换句话说，你可以使用CREATE SCHEMA或者CREATE DATABASE来创建一个数据库。

Oracle Database

在Oracle中，CREATE SCHEMA语句实际上并不创建一个模式，这是因为已经为在创建用户时，数据库用户就已经创建了一个模式，也就是说在ORACLE中CREATE USER就创建了一个schema，CREATE SCHEMA语句允许你将schema同表和视图关联起来，并在这些对象上授权，从而不必在多个事务中发出多个SQL语句。

SQL Server

在SQL Server中，CREATE SCHEMA将按照名称创建一个模式，与MySQL不同，CREATE SCHEMA语句创建了一个单独定义到数据库的模式。和ORACLE也不同，CREATE SCHEMA语句实际创建了一个模式(前面说到这个语句在ORACLE中不创建一个模式)，在SQL Server中，一旦创建了模式，就可以往模式中添加用户和对象。

总结

schema这个词可以用在很多不同的环境中，在特定数据库管理系统创建一个schema时，您需要使用DBMS特定定义模式，当你切换到一个新的数据库管理系统时，一定要查看该系统是如何定义schema的。

# [RBAC 介绍 (权限)](https://www.cnblogs.com/niuli1987/p/9871182.html)

[RBAC](http://www.sojson.com/tag_rbac.html)是基于角色的访问控制（Role-Based Access Control）在[RBAC](http://www.sojson.com/tag_rbac.html" \t "_blank" \o "RBAC)中，权限与角色相关联，用户通过成为适当角色的成员而得到这些角色的权限。这就极大地简化了权限的管理。这样管理都是层级相互依赖的，权限赋予给角色，而把角色又赋予用户，这样的权限设计很清楚，管理起来很方便。

# 常见的认证机制

1 HTTP Basic Auth

2 Cookie Auth

3 OAuth

OAUTH协议为用户资源的授权提供了一个安全的、开放而又简易的标准。与以往的授权方式不同之处是OAUTH的授权不会使第三方触及到用户的帐号信息（如用户名与密码），即第三方无需使用用户的用户名与密码就可以申请获得该用户资源的授权，因此OAUTH是安全的。oAuth是Open Authorization的简写。

4 Token Auth

支持跨域访问: Cookie是不允许垮域访问的，这一点对Token机制是不存在的，前提是传输的用户认证信息通过HTTP头传输

无状态(也称：服务端可扩展行): Token机制在服务端不需要存储session信息，因为Token 自身包含了所有登录用户的信息， 只需要在客户端的cookie或本地介质存储状态信息

更适用CDN: 可以通过内容分发网络请求你服务端的所有资料（如：javascript，HTML,图片等），而你的服务端只要提供API即可

去耦: 不需要绑定到一个特定的身份验证方案。Token可以在任何地方生成，只要在你的API被调用的时候，你可以进行Token生成调用即可

更适用于移动应用: 当你的客户端是一个原生平台（iOS, Android，Windows 8等）时，Cookie是不被支持的（你需要通过Cookie容器进行处理）， 这时采用Token认证机制就会简单得多。

CSRF: 因为不再依赖于Cookie，所以你就不需要考虑对CSRF（跨站请求伪造）的防范。

TOKEN 认证过程：

客户端使用用户名跟密码请求登录 服务端收到请求，去验证用户名与密码 验证成功后，服务端会签发一个 Token，再把这个 Token 发送给客户端 客户端收到 Token 以后可以把它存储起来，比如放在 Cookie 里或者 Local Storage 里 客户端每次向服务端请求资源的时候需要带着服务端签发的 Token 服务端收到请求，然后去验证客户端请求里面带着的 Token，如果验证成功，就向客户端返回请求的数据 因为token是被签名的，所以我们可以认为一个可以解码认证通过的token是由我们系统发放的，其中带的信息是合法有效的