Meeting with Steve, Wayne, John, Alan, and Brandon

8/18/22

* How can we make each corp stand on its own database – corp-specific
* Wayne was pitching using the corp folder hash to name the database schema
* The main login is the only place where we need to make decisions
* Question about the payee table… currently it is showing both users and vendors
* Universe level
* Cluster level
* Wayne wants to create a user type table – people who use the system
  + Maybe having both a user record and a payee record.
  + Payees are just for people who will be paid
  + We will need a master list
* What about a centralized OAuth2 based login
* Wayne was talking about using a master list – out on Amazon – and then return them to the local server and setup the session stuff.
* We have 20+ data servers – Ideally, we would love a single login and then allow the user to bounce wherever they have permissions. This could be login to data 10, then bounce to data 20, then back to 10, then out to 0. Depending on permissions and access levels.
* What about cross corp transactions?
* Keeping track of who goes and accesses what – Where did so and so go and what did they touch?
* We could setup nightly services that keep things up to date.
* At some point, we will need some super admin type tools – universe level permissions and tools – they could do almost anything and help and manage things.
  + We may need permissions there as well.
  + Global permission and corps
* If we run reports that cross corps (we would need all of those database schemas on the same server – but they would be their own independent schemas) we would have to recode things and do some Unions.
* DSN – datasource names
* If we get into crazy cross corp and cross server stuff – we could use API sockets and build temp tables or temp databases to bring things together. This is more complex and is not as fast.
* As we make these changes, we will be touching every query in the entire system.
* Load balancing will be easier. Updates may be more complicated.
  + If doing updates on corp-specific tables, there is more clicking
  + Wayne is proposing a database update service - asynchronous type updates
    - We could push things to the service and then it does the update
    - We start something and then it does it in the background vs us touching each piece or section
    - Database updates have been somewhat of a nightmare
    - Wayne is doing some version control stuff in his settings services. There is also some code in the database update service. Different ways to keep track of what is needed and what is currently running.
    - They are using the Application.cfc – on start up, on page request, on session start, on application start up, etc.
* Certain tables will be standard (shared or synced) and others will be totally corp-specific.
* Brandon would like to keep to the solar system type terminology
* John was saying… think of the Sun – This is the database server – Then all of the worlds (corps) would revolve around the sun. This is kinda like a solar system.
* Wayne – proposing things – having myCorpId and myUserId as variables. This is who you are and where you are logged in.
  + Let’s not use current – that is alrealy all over the system
  + What about the word “targe” or “active” – Let’s use target. What am I trying to access. targetCorpId and targetUserId
  + Being consistent to manage those pieces.
* We have to get this stuff in the docs – everybody needs to know and play along accordingly
* We may need certain things to sync between servers. This could be master user lists, global user lists, permissions, etc.
  + There are tons of other table that need to be synced up
* Wayne wants to simplify it
* Every user will have a global user id number
* On the main login – eventually, we wanted to do a concept of the adilas café – a single login, we check to see where you can go, then you bounce to those areas (worlds or corps).
  + A single login in one place and then we can push or allow the users to jump where they need to.
  + Maybe use a form of tokens to validate logins
  + If we used tokens and validation – we could jump between servers without having to login to each server. The token, if active, allows the jumping and then we help auto login the user without them inputting the usernames and passwords.
* As a reminder, we started making some new tables that all have the name “cluster\_[something]”. There may be some ideas there.
* We have to keep track of users who live and die in one corp (world) and we have others who will be bouncing between corps (worlds), servers (clusters or solar systems), etc. It could be super simple or it could be a virtual universe level traveler
* Light talk about URL’s and web addresses – Ideally, we have a single login and then use the tokens to bounce wherever needed. That could be a new world (corp) on the same server or it could be a complete jump to a new server (possible URL change).
* Other options – dealing with mirrors and ways of pushing production data out to other database boxes.
  + Even possible universe or enterprise level of data warehousing and aggregating
* Wayne was also talking about config type files that could be held outside of Git and bit bucket (special settings as needed).
* Right now, we only get to point to one DSN (datasource name). If we can break that out… it opens up tons of options, including production data and testing data. Lots of possible options.
* Steve – we want to be quick and fast but we can’t keep everybody on the same server
  + We are thinking a global login (one computer or server)
  + Then use different data servers
* Wayne would like to work on clustering the database server. Basically, one super computer that does all of the database work (all with smaller corp-specific databases) and works in a cluster type environment.
* We have had customer complaints about cloudflare and having to pick busses, boats, or motorcycles to validate that they are a human vs a bot.
* As a huge side note, we have really gained by having some of the servers separated. If something goes down or wrong, it doesn’t affect anybody else. That has been huge.
* If we decide to cluster things, we have a few options.
  + We could cluster the database stuff easier than clustering the data (code or ColdFusion) servers or web servers.
    - We just converted the database engine to Innodb tables vs the older MyISAM tables. That will allow us to cluster things better.
    - On the web or data servers, we still have problems with sessions, files, and local assets (files, images, PDF’s, excel files, CSV’s, etc.)
* Alan was talking about horizontal and vertical scaling
* This will be a process – we need to communicate as we go – the whole team.
* We have never done any clustering. We are heading into that area and environment.
* Wayne is proposing that we could break things out a little bit at a time. Cherry pick things – add in schema names and then table names.
* Wayne and Alan were talking about versions and even flip flopping between testing and live. There have been lots of problems with init functions, setting up the first instance of objects and services, etc.
  + Alan was asking about garbage collection and memory management stuff
* We’ve got to get things into the adilas docs – standardizing things for all of our developers
  + Naming conventions
  + Types
  + Classes, services, vars, methods, database fields, etc.
  + Refactoring some of the classes and services – on going – trying to standardize things
  + Organizing things as we touch them
  + Separating queries out of the code and refactoring things where things are better organized
    - Say 100 lines of code down to 10 lines of code
    - Figuring out queries that are duplicated over and over again
    - Reusing things and being efficient
* Cherry picking what we want to change – the art of becoming… Small bites and smaller changes – easier to manage
* Managing burnout
* Wayne would like to pick some tables and start breaking things into the smaller breakouts and getting certain tables into their own schemas. Tables like zip codes, other look-up tables, and underlying supporting tables. Basically, some of the static tables.
* Eventually we want to bill our customers based off of what they are storing… We have a number of things that we could start watching to help with billing for data, storage, and usage.
* Wayne already has some data schema copying code written – with some minor tweaks, it is ready.
* When we create a new corp (world) – it will need to do some dancing – all via scripts that are being ran asynchronously. Threads and helping to use progress and timing.
* Our new current standard is Innodb for the table engine, utf8mb4 for the character set, and the collation general ci (ci stands for case incentive)
  + Sample: CREATE TABLE `zipcodes` (  
    `zip\_id` int(10) unsigned NOT NULL AUTO\_INCREMENT,  
    `zipcode` varchar(15) DEFAULT NULL,  
    `city` varchar(100) DEFAULT NULL,  
    `county` varchar(100) DEFAULT NULL,  
    `state` varchar(100) DEFAULT NULL,  
    `state\_fips` varchar(100) DEFAULT '',  
    `state\_code` varchar(4) DEFAULT NULL,  
    PRIMARY KEY (`zip\_id`)  
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW\_FORMAT=DYNAMIC;
* John is going to setup and do some work on the docs
  + Alan is going to pass some info to John on getters and setters
* As we do some merging… we need to start checking naming conventions and other standards. This will be on ongoing effort.