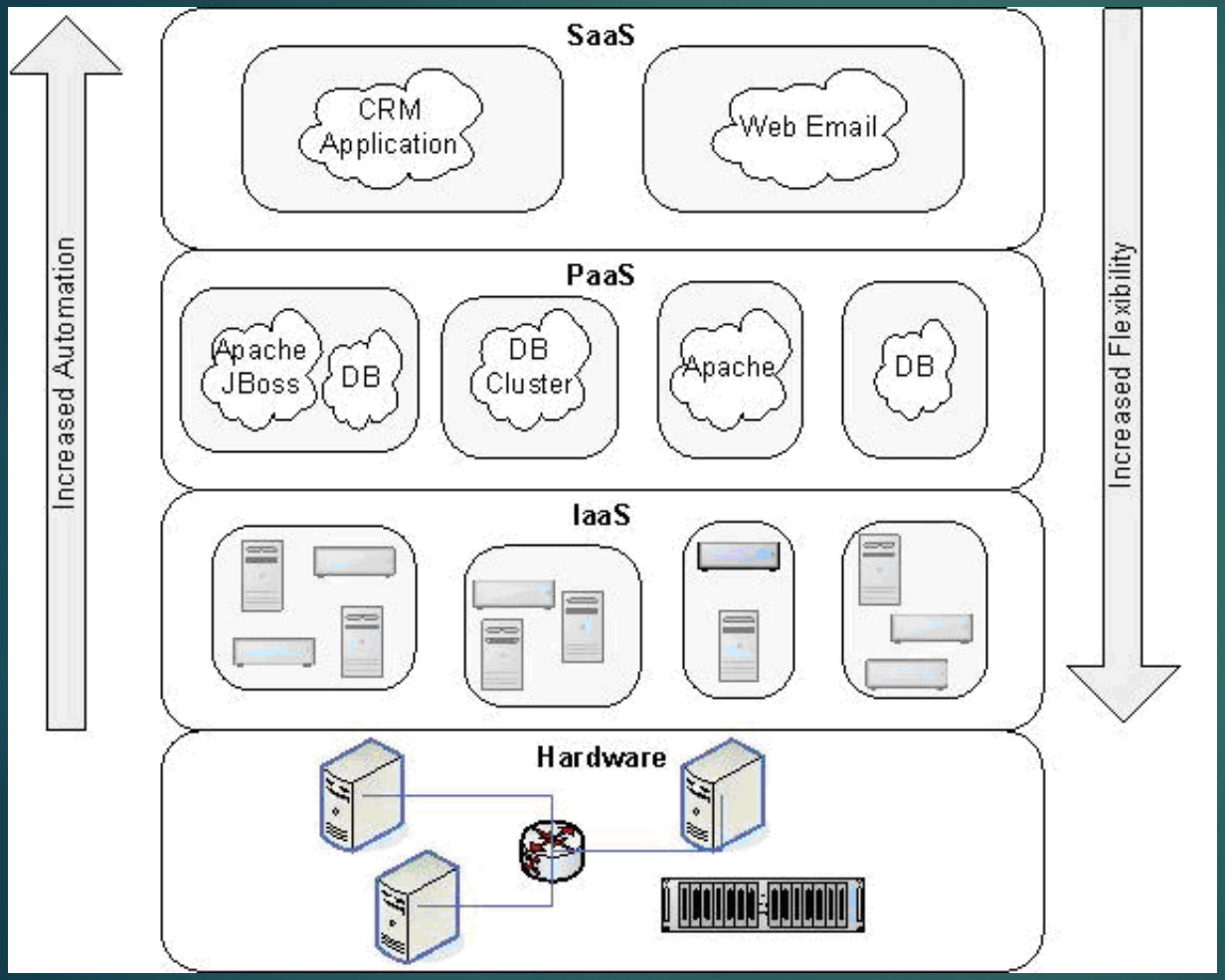
Software As A Service and Multitenancy

K V SUBRAMANIAM

Background

Comparison: Cloud Service Models

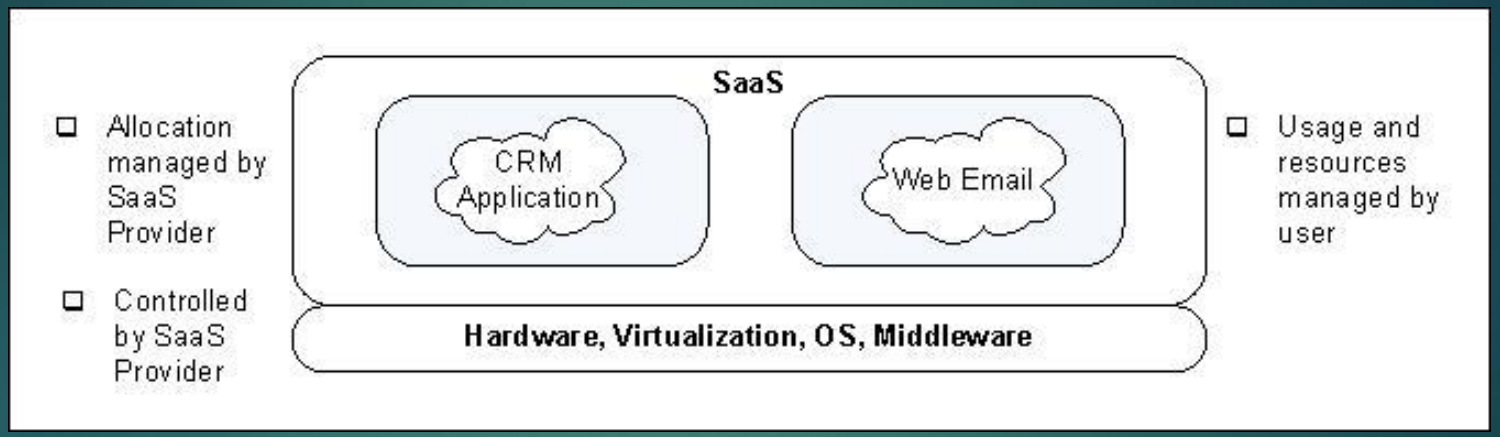


Software As A Service (SaaS)

Platform As A Service (PaaS)

Infrastructure As A Service (IaaS)

Software as a Service (SaaS)



NIST definition: The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser.

SaaS Benefits

Characteristics	Benefits
Network delivered access to commercially available software	**No local infrastructure or software to purchase or maintain** Applications & data are available anywhere with network connectivity
Application delivery is one-to-many model	Operating costs are reduced by managing infrastructure in central locations rather than at each customer's site
Built on optimized & robust platform	Improved availability and reliability
Customer pays for as much as they need when they need it	Lower TCO

Exercise 0: Revenue Options

- Subscription (monthly rental)
- Freemium model – starts free and then increases price
- Tiered pricing
- Usage/transaction based
- Per user pricing
- Per feature pricing
- Ad-based revenue (pay per click)
- Profit sharing
- Ownership Sharing

- For each of the options on the left, name one SaaS company that does this

SaaS architecture

- Front-end or GUI for SaaS application is a browser
- Business Logic runs on the backend cloud infrastructure
- Business Logic frequently implemented as *microservices*
- Examples: Google Docs, Gmail, Salesforce.com

Exercise 1

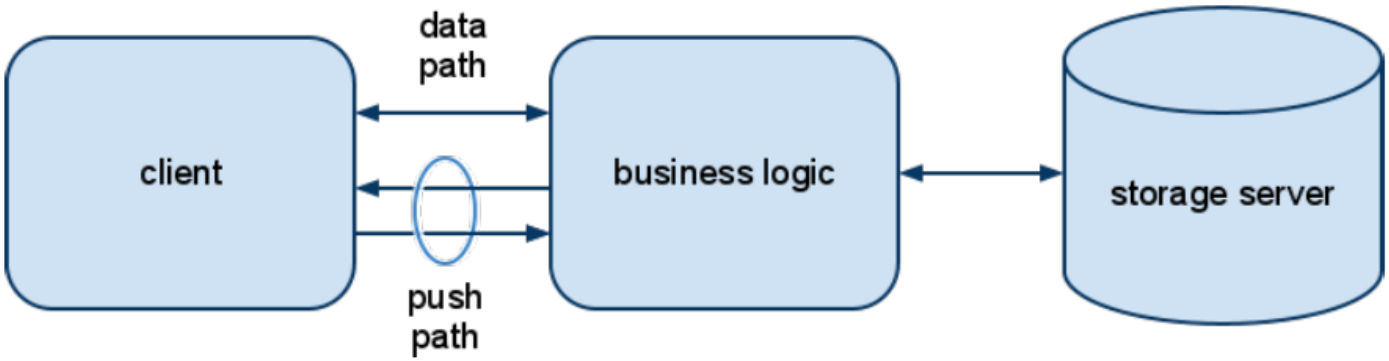
- You have built a Bookstore application - which permits users to login to a portal to query and purchase books
- What UI challenges will you have to face?

Solution GUI

- Need screens similar to Amazon for each microservice
- Challenges
  - Ensure that the app runs well on all browsers
  - Handling performance lags

Case Study: Gmail

Macro-architecture



data path: code, styles, preferences, user data, ...
push path: chat, new mail, presence, contact updates, buzz updates, ...

Google

Courtesy: Adam De Boor: Gmail: Past, Present and Future

Gmail: Client (Front-end) Functions

- Look and feel
  - Builds all UI based on browser type
  - Loads GUI code (Javascript) as needed
  - Widgets: Provides uniform look/feel to different windows
- Performance
  - Fetches and caches data
  - Records actions for GUI performance analysis
    - Determine round-trip time for a 1pixel image
- Reports presence/idle to back end
- Drives multiple windows

Courtesy: Adam De Boor: Gmail: Past, Present and Future
https://dzone.com/articles/gmail-architecture-part-1

Exercise 2 (10 mins)

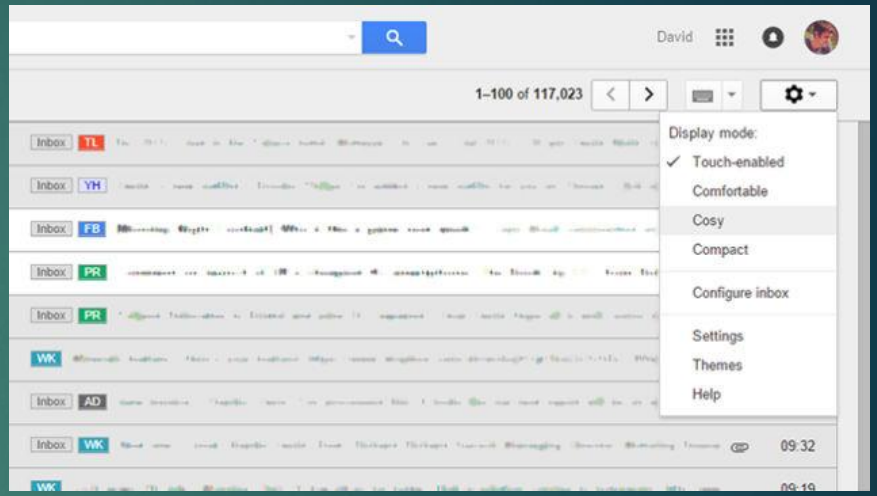
- For the bookkarts application which permits users to login to a portal to query and purchase books
- Pustakhouse buys this application
- Approached by another bookstore – ClassyBooks that wants additionally
  - Name + Front cover + review of books to be displayed
  - Different layout of the page
  - Customers to be able to make one click payment
- How will you customize the GUI?

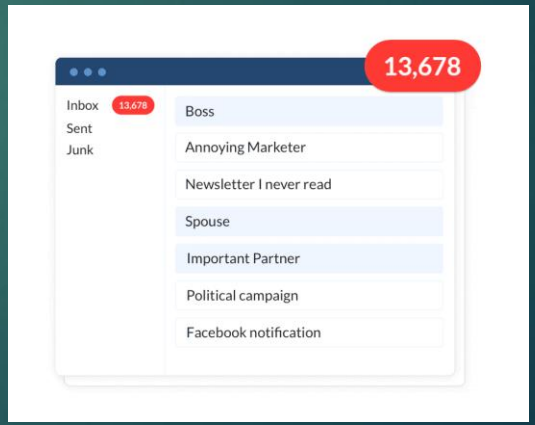
Solution: GUI Screens

- How to customize the screen?
  - Typically we need one GUI for the app as used by the customers of the SaaS application
    - The business application
  - Another one for customizing the App for the vendor for use by the vendor
    - Used for defining the layout, logos

Case Study: Gmail Customization

- Settings and Layout
- Gmail Themes
- Third party add ons
  - Sane-Box





Further Reading – Details of Gmail architecture

Exercise 3 (10 mins)

- For the bookkarts application which permits users to login to a portal to query and purchase books
- Pustakhouse buys this application
- Approached by another bookstore – ClassyBooks that wants additionally
  - Name + Front cover + review of books to be displayed
  - Different layout of the page
  - Customers to be able to make one click payment
  - Different schemas?
- How will you design the backend?

Exercise 3 – Backend Design

- Name + Front cover + review of books to be display
- Customers to be able to make one click payment
  - Requires different workflows
    - Allow users to customize workflows
  - Separate functionality into microservices – RESTful services
- Different layout of the page
  - Allow GUI customization
- Different schemas
  - Multitenancy.

See details about Backend Design in the Further Reading section

Multi-Tenancy

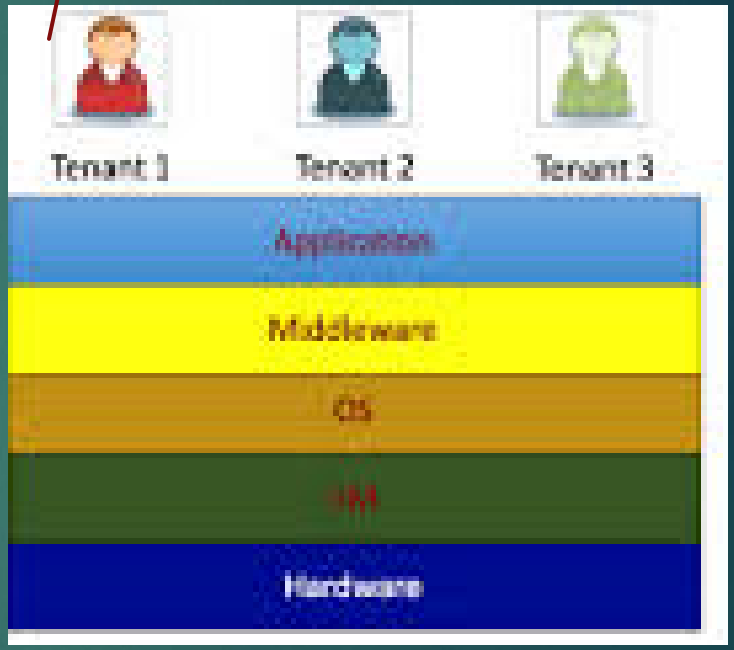
Multi-tenancy vs Virtualization

- Virtualization: coarse-grained resource sharing
  - Generally hardware, difficult to do fine-grained sharing
  - Compute / cores
  - Memory
- Fine-grained resource sharing, generally software

Terminology

- Tenants
  - Customers of the cloud provider
  - Organizations or users
- Users
  - Users of the cloud system
  - May belong to a tenant

Think of tenant as a family that rents an apartment and its services like gym etc.



Users are people that stay in an apartment.

Motivation

- Experiment by Jacobs et al.
- 3 methods for creating a multi-tenant database
- Shared machine
  - Db process and tables per tenant
- Shared process
  - DB tables per tenant
- Shared table
  - Process and tables shared

- Shared machine
  - 55 MB main memory, 4MB disk memory per tenant
- Shared process
  - 80 MB main memory for 10,000 tenants
- Challenges
  - Security
  - Customization
    - Different schema

Jacobs, D, Aulbach, S, Marz 2007. Ruminations on Multi-Tenant Databases. 12.GI-Fachtagung fur Datenbanksysteme in Business, Technologie und Web (BTW 2007), 5 bis 9, Aachen, Germany.

Multi-tenancy Levels - Summary

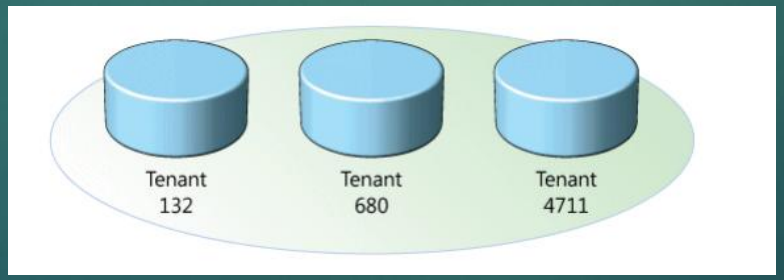
- Different types of multitenancy possible
- Simplest - each client has own version of software
- Most complex - single version of software hosted on a cluster. Supports multiple tenants.

Further Reading section – for details

https://renatoargh.wordpress.com/wp-content/uploads/2018/01/article-multi-tenant-data-architecture-2006.pdf

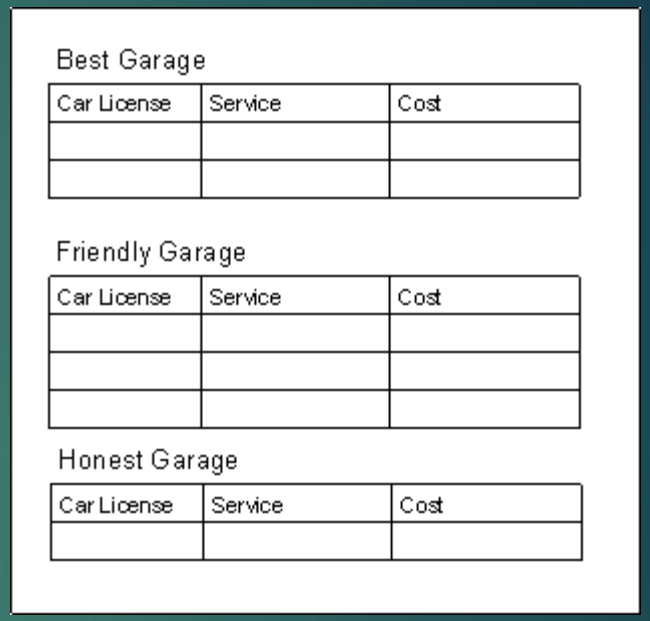
Storage Multi-tenancy

- File systems already have well-known methods for sharing
- Database methods
  - Dedicated tables per client
  - Shared tables per client



Dedicated Tables per Tenant

- Each tenant has their own tables
- Shared table, separate schema
- Generally DBs store different tables in different files
  - Possibly greater security
- Details
  - Register the three garages as DB users
  - `GRANT SELECT ON FriendlyTable TO FriendlyGarage WITH GRANT OPTION`



Best Garage

Car License	Service	Cost

Friendly Garage

Car License	Service	Cost

Honest Garage

Car License	Service	Cost

Shared Table Option - Basics

- All data stored in the same table
- Each row has a TenantId which identifies the tenant to which the data belongs
- More space efficient, but may need an additional SELECT
- Can use a key to encrypt the data for each tenant
- Metadata table to store information about the client

Data Table 1

Tenant Id	Car License	Repair	Cost
1			
2			
2			
1			
3			
2			

Metadata Table 1

Tenant Id	Data
1	Best Garage
2	Friendly Garage
3	Honest Garage

Shared Table Option - Customization

- Customization
  - Different tenants would want to define their own custom fields
- How can this be done efficiently?

DB Customization – Pre-allocated Columns

- A number of pre-defined custom columns in each row
  - Salesforce has 500
- Metadata table stores the type and name of each column

Data Table 1

Tenant Id	Car License	Service	Cost	Custom1	Custom2
1					
2					
2					
1					
3					
2					

Metadata Table 1

Tenant Id	Tenant Name	Custom1 Name	Custom1 Type
1	Best Garage	Service Rating	int
2	Friendly Garage	Service Manager	string
3	Honest Garage		

Exercise 4 (5 minutes)

- Consider the table to the right
- Consider a query from *BestGarage*
  - *SELECT CarLicense, ServiceRating FROM CustomerTable WHERE ServiceRating > 8;*
- What
  - Algorithm would be used by the compiler to translate this into a query on *DataTable1?*
  - What is the resulting query?

Data Table 1

Tenant Id	Car License	Service	Cost	Custom1	Custom2
1					
2					
2					
1					
3					
2					

Metadata Table 1

Tenant Id	Tenant Name	Custom1 Name	Custom1 Type
1	Best Garage	Service Rating	int
2	Friendly Garage	Service Manager	string
3	Honest Garage		

Exercise 4 (5 minutes)

- Consider query
  - *SELECT CarLicense, ServiceRating FROM CustomerTable WHERE ServiceRating > 8;*
- Algorithm
  - Check from *Metadata Table* that *ServiceRating* field exists for *BestGarage*
  - Find the corresponding column from *MetadataTable 1* as *Column 1*
  - Add an additional condition to *WHERE* clause
  - *SELECT CarLicense, int(Custom1) FROM CustomerTable WHERE (int(Custom1) > 8) AND (TenantId=1);*

Data Table 1

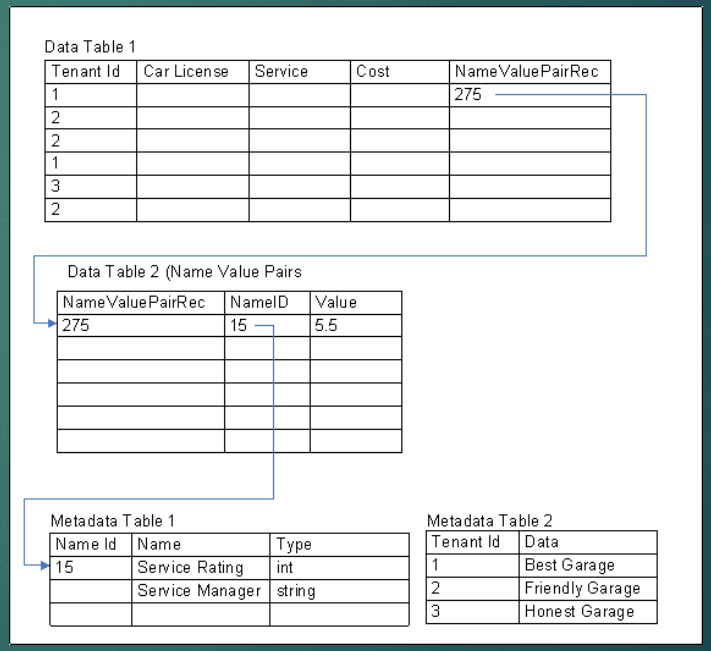
Tenant Id	Car License	Service	Cost	Custom1	Custom2
1					
2					
2					
1					
3					
2					

Metadata Table 1

Tenant Id	Tenant Name	Custom1 Name	Custom1 Type
1	Best Garage	Service Rating	int
2	Friendly Garage	Service Manager	string
3	Honest Garage		

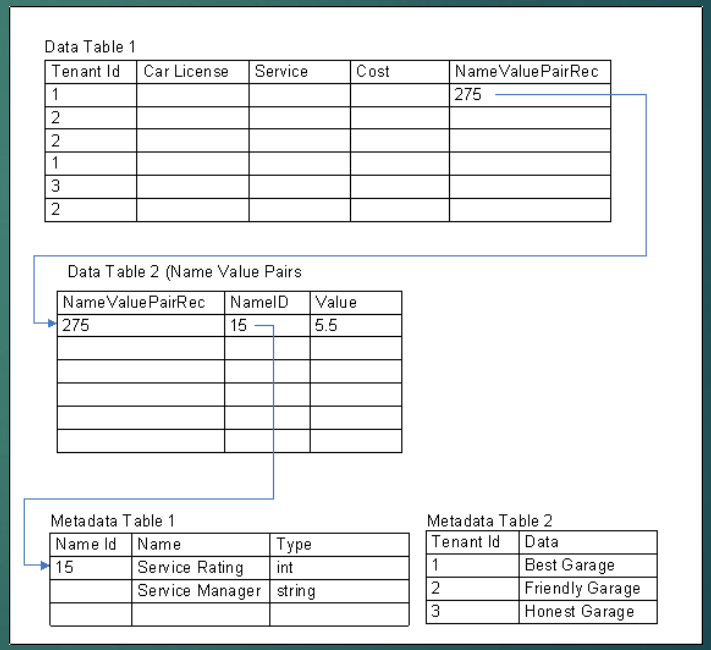
DB Customization – Name-value Pairs

- Main table has a name-value pair record column
- Pivot table has list of name value pair
  - If there is more than one name-value pair, there will be multiple records in the pivot table
- Name is actually an id that points to the actual name and type in a metadata table
- Greater space efficiency; however joins may be needed to re-construct records



Exercise 5 (15 minutes)

- Suppose *BestGarage* has an additional custom field *MechanicId* of type integer
  - How would *DataTable2* and *MetadataTable1* change?
  - Repeat previous exercise for *BestGarage* query
    - *SELECT CarLicense, MechanicId, ServiceRating FROM DataTable WHERE ServiceRating > 8*



Data Table 1

Tenant Id	Car License	Service	Cost	NameValuePairRec
1				275
2				
2				
1				
3				
2				

Data Table 2 (Name Value Pairs

NameValuePairRec	NameID	Value
275	15	5.5

Metadata Table 1

Name Id	Name	Type
15	Service Rating	int
	Service Manager	string

Metadata Table 2

Tenant Id	Data
1	Best Garage
2	Friendly Garage
3	Honest Garage

Scaling in SaaS

- Built on PaaS
- So, all PaaS based scalability is relevant to SaaS
- Additionally
  - Multi-tenancy for improved scaling/utilization
  - Query optimization – per tenant in real time
    - <300 ms latency

SaaS Summary

- SaaS based on the following principles
  - Multi-tenancy
  - Customization
  - Scaling