# Tutorial – PaaS

## Case Study: Force.com application development platform

Week 6

School of Software

Faculty of Engineering and Information Technology

University of Technology Sydney

SCHOOL OF SOFTWARE

# Short review of PaaS

- Overview of PaaS
  - Composed of hardware resources and software resources (such as operating systems and/or software frameworks) to ***build*** software applications;
  - Could be provisioned on top of IaaS;
  - Shared hardware resources and software resource usage among all PaaS consumers;
  - Major PaaS providers: Google App Engine, Force.com platform, and Microsoft Azure.

# Case study for Force.com platform

- **Question 1:** What are multi-tenant applications?
- **Question 2:** What are the benefits of multi-tenant applications?
- **Question 3:** Architecture overview of Force.com
- **Question 4:** Data storage model of Force.com

# Case study for Force.com platform

- **Question 1:** What are multi-tenant applications?

  (refer to Section 2)

- **Question 2:** What are the benefits of multi-tenant applications?

  (refer to Section 2 and 3)

- **Question 3:** Architecture overview of Force.com

  (refer to Section 4 and 5)

- **Question 4:** Data storage model of Force.com

  (refer to Section 6)

# Q1: What are multitenant applications?

- Multitenant applications satisfy the needs of multiple users or organizations (in contrast to single tenant applications).

- In PaaS multi-tenant applications, all tenants share the same underlying computing platform

- Tenants operate in virtual isolation from one another, and their data and customizations remain secure and insulated from the activity of all other tenants.

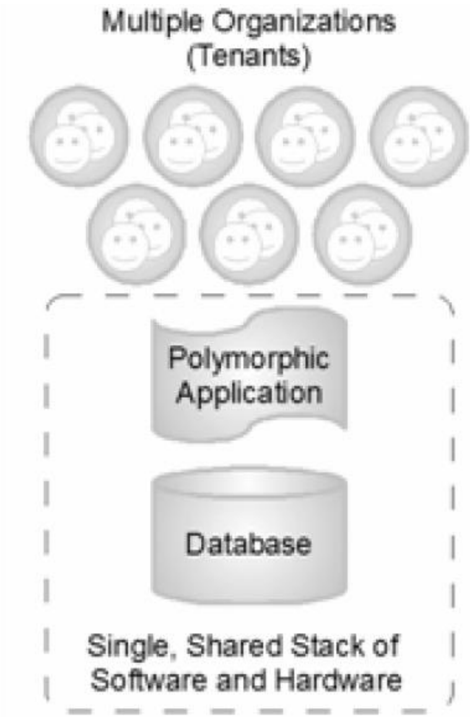

Multiple Organizations (Tenants)
Polymorphic Application
Database
Single, Shared Stack of Software and Hardware

**Figure:** Tenants sharing a single stack of computing resources

# Q2: What are the benefits of multi-tenant applications?

- **Benefits to application providers:**
  - Ease of carrying out application update(s);
  - Enabling and managing application delivery for multiple users using a single stack of resources (much easier and cheaper than maintaining a single stack of resources for each user);
  - Provider gather operational information from the collective user population (which queries respond slowly, what errors happen more often, etc.)
- **Benefits to the end users:**
  - Receive the applications at a (much) lower price than traditional mechanisms

# Q3: Working of Force.com (at an abstract level)

- Core-application data and logic is stored as metadata

- End-user customizations (changes to existing core data, core logic, new data, and new logic), are stored as metadata (and not as views).

- Force.com provides clear separation between:
  (a) Core application data and functionality (Force.com defined Data and Logic)
  (a) User-defined application data and functionality (Custom Data and Logic)

- Force.com does not create an "actual" table in a database (in response to customizations), rather it's runtime engine generates these views on the fly using the metadata

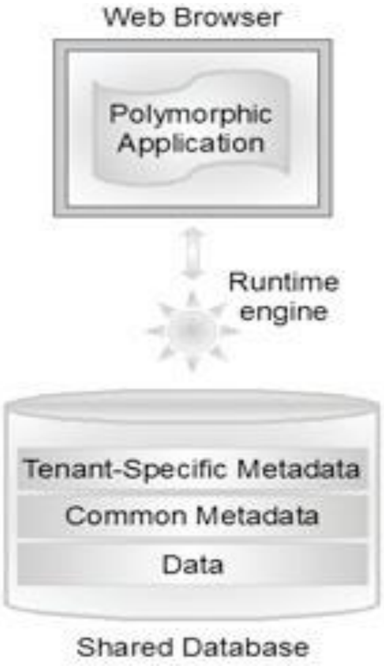


**Figure:** Force Multi-tenant Architecture

- Different users (tenants) see their customized (tenant-specific) view of the application

# Q3: Architectural elements of Force.com

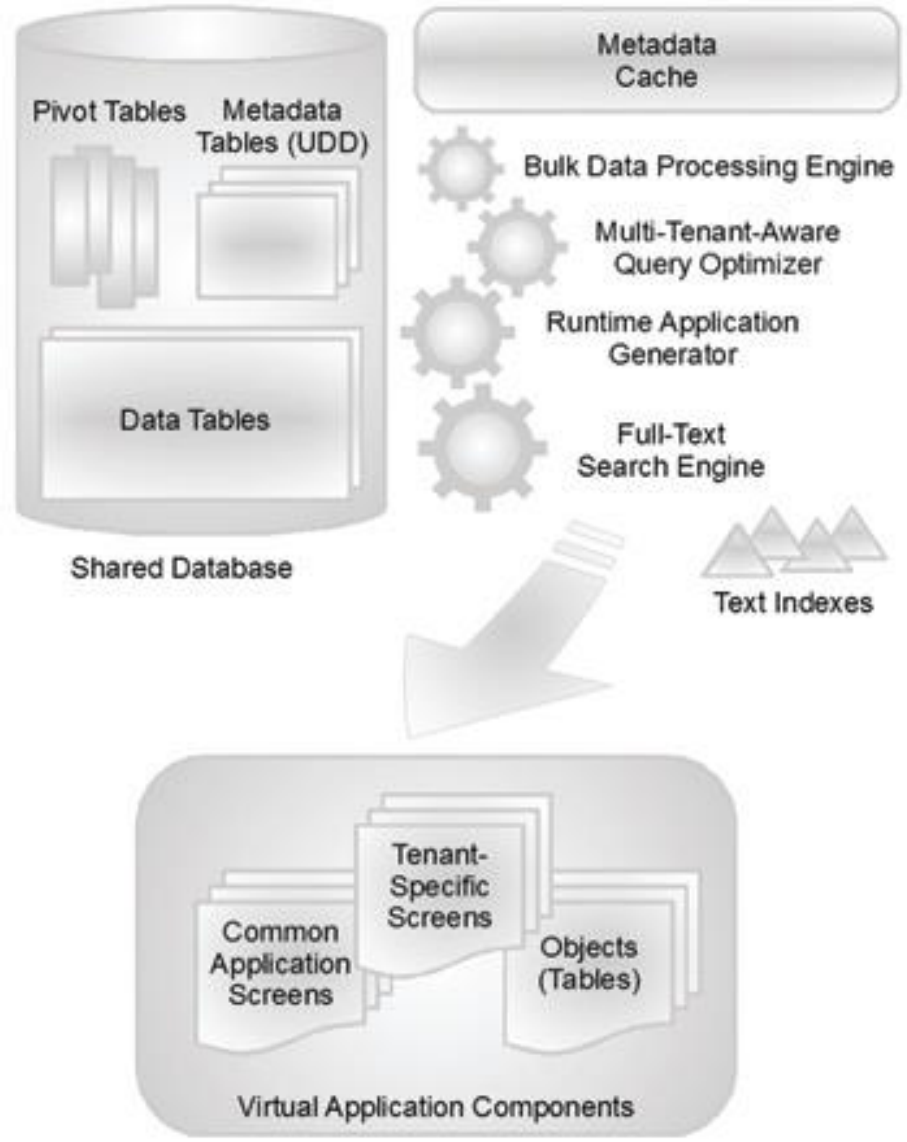


**Figure:** Architectural elements of Force.com platform

# Q3: Architectural elements of Force.com

- Metadata cache:
  - Frequently accessed metadata (used to generate application components) is stored in metadata cache;
  - This improves performance and application response time.

- Bulk Data processing engine:
  - This engine carries out repetitive actions (such as bulk update, bulk retrieve, bulk delete etc…);
  - Has an intelligent in-built fault-recovery mechanism to re-execute the “update” or “save” operations after factoring out records that cause errors during the bulk update/save process

# Q3: Architecture overview of Force.com

- Full text external search engine:
  - Force.com employs an external search service
  - This works independent of the other architectural elements
  - Tenant’s data is indexed (and indexes are stored as a special set of pivot tables) for quick access and retrieval.
- Runtime Application Generator:
  - Runtime engine accesses the
    - core application metadata;  (and)
    - custom metadata of the corresponding user.
  - Runtime engine compiles them at run-time to generate custom views for the corresponding user

# Q3: Architecture overview of Force.com

- Multitenant-aware query optimizer
  - The underlying storage tables are partitioned based on Org ID
  - The query optimizer, considers which users is executing a given application function, and using intelligent algorithms determines which database partition needs to be queried

# Q3: Architecture overview of Force.com

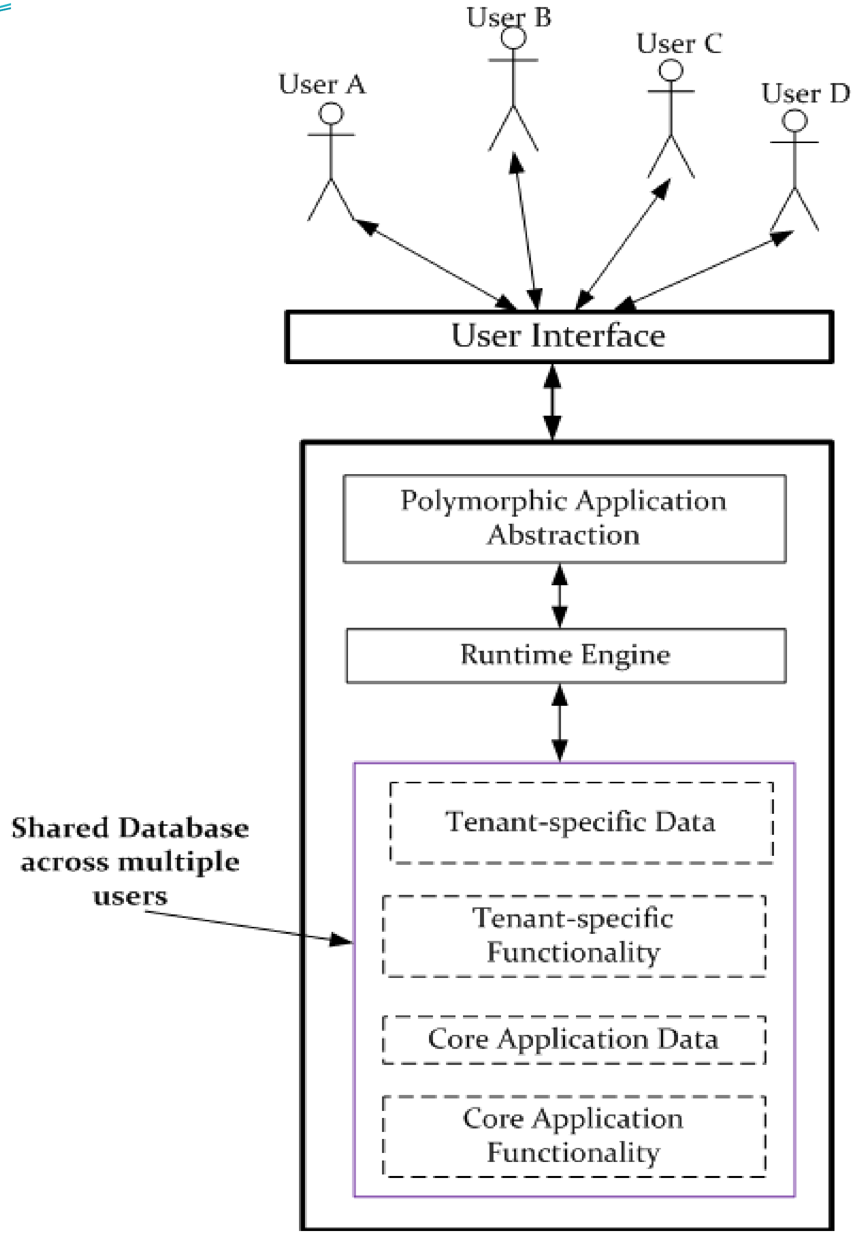

User A
User B
User C
User D
User Interface
Polymorphic Application Abstraction
Runtime Engine
Tenant-specific Data
Tenant-specific Functionality
Core Application Data
Core Application Functionality
Shared Database across multiple users

**Figure:** Step-wise working of Force.com

**Step 1:** A given user (assume User A) interacts via a user interface and provides authentication credentials

**Step 2:** The user interface passes User A's credentials and/or request for information from User A to the Polymorphic Application Abstraction Layer

**Step 3:** The polymorphic application abstraction determines the user requesting the data and the type of data that is being requested and passes this information to the run time engine

**Step 4:** The run-time engine takes input from the polymorphic application abstraction, retrieves relevant data, compiles it, and passes it back.

# Q4: Data Storage Model of Force.com

- Force.com manages to deliver the views (including the customized user defined views) using a combination of:
  - Metadata tables
  - Data tables
  - Pivot Tables

- Joins between the above three tables are carried out at run-time to generate customized views.

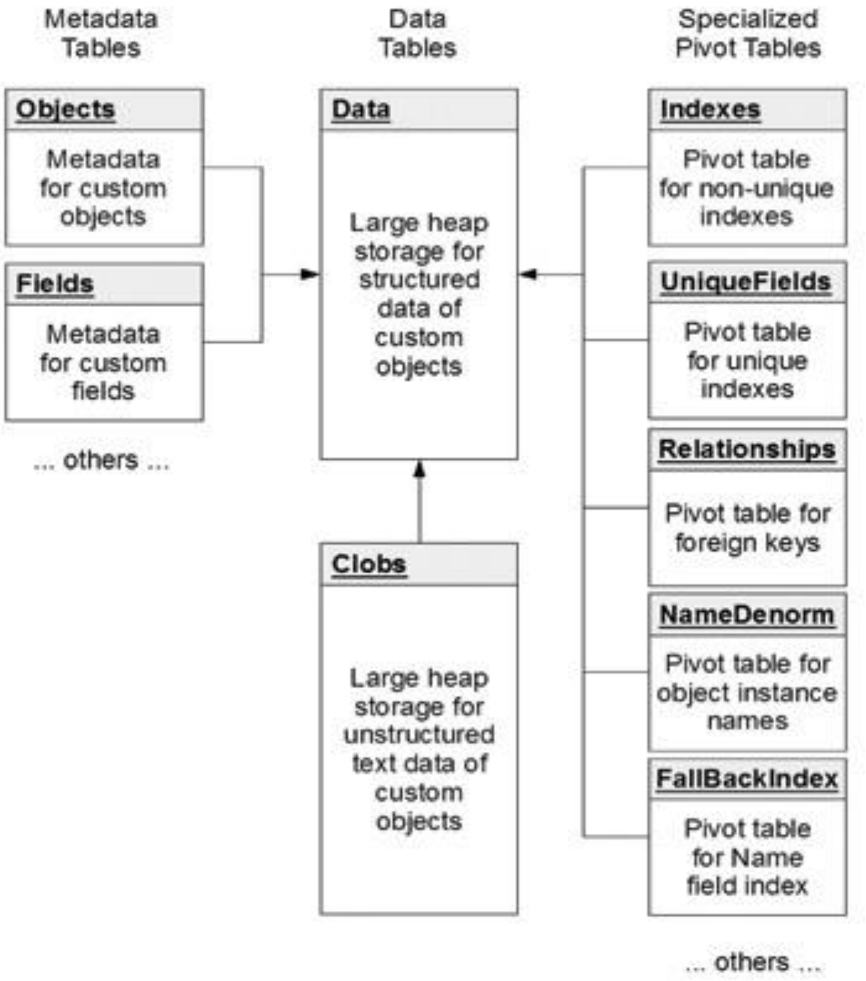
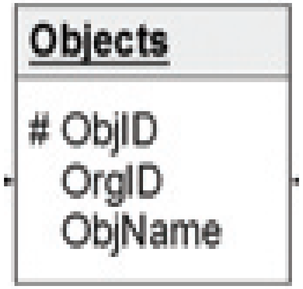

Metadata Tables
Data Tables
Specialized Pivot Tables
Objects
Metadata for custom objects
Fields
Metadata for custom fields
... others ...
Data
Large heap storage for structured data of custom objects
Clobs
Large heap storage for unstructured text data of custom objects
Indexes
Pivot table for non-unique indexes
UniqueFields
Pivot table for unique indexes
Relationships
Pivot table for foreign keys
NameDenorm
Pivot table for object instance names
FallBackIndex
Pivot table for Name field index
... others ...

**Figure:** Force.com data storage model

# Q4: Data Storage Model of Force.com
## Metadata table – Objects Metadata table

- Two types of metadata tables to store user's customizations
- Objects Metadata table
  - Used to store "*information*" (not the corresponding data values within the object), about the user-defined objects.
  - Stored object information is annotated with
    - OrgID - A unique identifier assigned to each unique end-user
    - ObjID - Force.com assigned unique identifier to each user-created object
    - ObjectName – The name of the object
  - Each new user-defined class  will have an associated row in this table



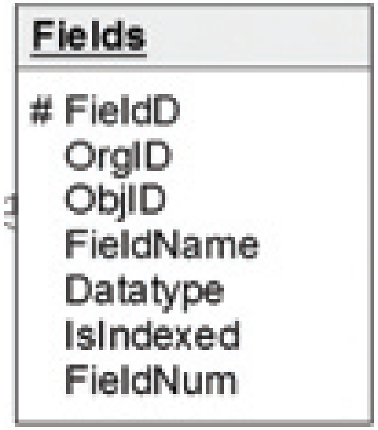
Objects

# ObjID
OrgID
ObjName

**Figure:** Metadata table for the storing information on custom classes

# Q4: Data Storage Model of Force.com
# Metadata tables – Fields Metadata table

- Fields Metadata table
  - Used to store "*information*" (not the corresponding data values) about the user-defined fields within an object.
  - Stored field information is annotated with:
    - OrgID - The identification of the end-user who owns the object encompassing the field
    - ObjID - The identification of the object to whom the field belongs
    - FieldName – The name of the field
    - IsIndexed – Boolean value to indicate if the field requires indexing
    - FieldNum – Value to determine the position of the field in the object relative to other fields
  - Each new user-defined field within an object will have an associated row in this table



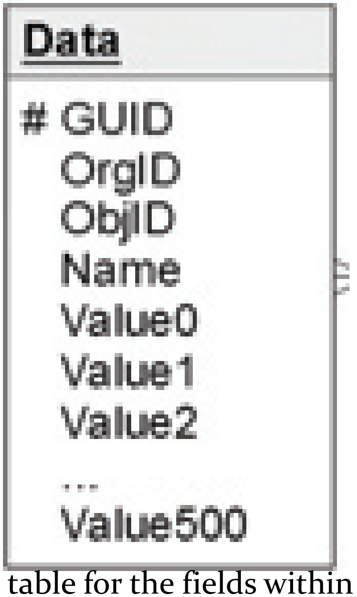
**Fields**

# FieldD
OrgID
ObjID
FieldName
Datatype
IsIndexed
FieldNum

**Figure:** Metadata table for the fields within an object

# Q4: Data Storage Model of Force.com - Data table

- Data table used to store the application-specific data
- The stored data is annotated by the following identifiers:
  - OrgID, ObjID, ObjName, and Global Unique Identifier (assigned by Force.com to the corresponding data)
- The type of the attributes in the data table is Custom/Flex, to accommodate any structured data types supported by the Force.com platform
- Each new user-defined object will have an associated row in this table



Data

# GUID
OrgID
ObjID
Name
Value0
Value1
Value2
...
Value500

**Figure:** Data table for the fields within an object

# Q4: Data Storage Model of Force.com CLOBs tables

- Force.com platform supports data storage as "Text Area (Long)" up to 32, 768 characters. This is used to store arbitrary and unstructured text.
- If an object has "Text Area (Long)" as one of its fields, then the corresponding data is stored in the CLOB (Character Large Objects) table

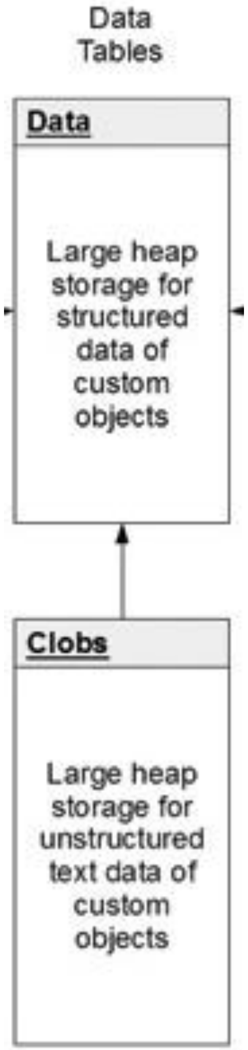
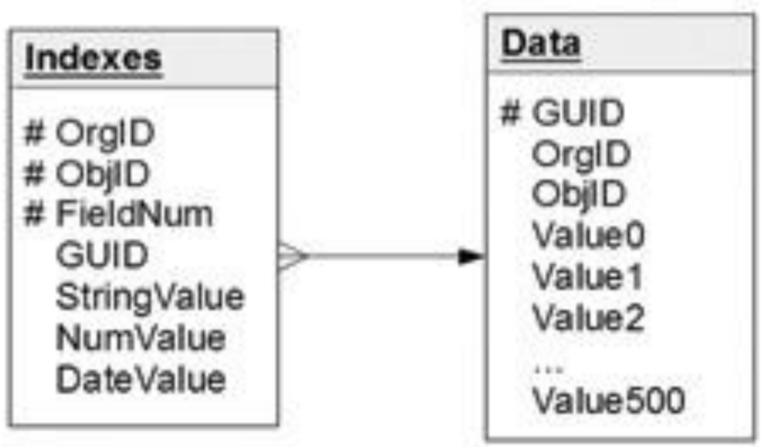

Data Tables
Data
Large heap storage for structured data of custom objects
Clobs
Large heap storage for unstructured text data of custom objects

**Figure:** Text Area (Long) fields are stored in CLOB table

# Q4: Data Storage Model of Force.com
## Pivot tables

- The external search engine indexes user’s data and stores its summary (based on different criteria), in the pivot tables .
- Indexes Pivot Table
    - This table indexes strongly typed data types with in the Force.com platform to enable fast searching on these data types



Indexes

# OrgID
# ObjID
# FieldNum
GUID
StringValue
NumValue
DateValue

Data

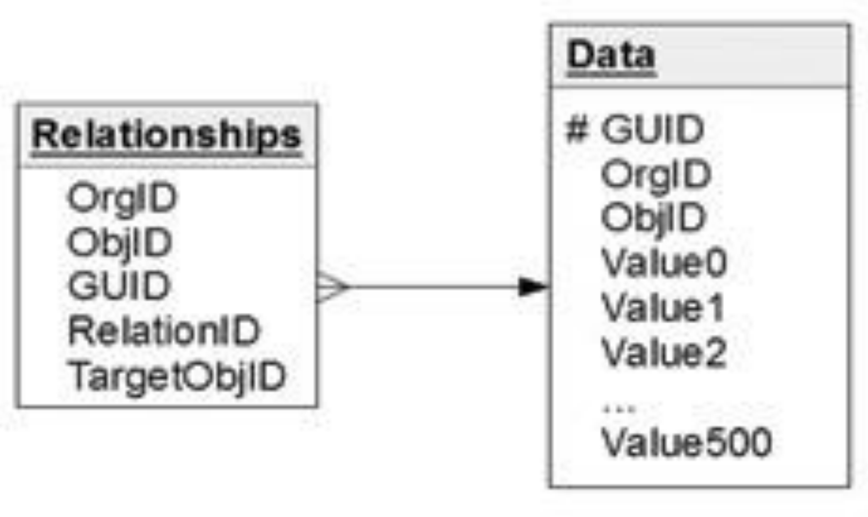
# GUID
OrgID
ObjID
Value0
Value1
Value2
...
Value500

**Figure:** The index pivot table stores a pivot of the indexed object field

# Q4: Data Storage Model of Force.com Pivot tables

- UniqueFields Pivot Table
  - This table is used to enforce the uniqueness of a user-defined field (if defined as a unique field within an object by the user)

- Relationships Pivot Table
  - This table stores a summary of all the relationships between custom objects;
  - Custom-relationships cannot be defined (at this stage) in Force.com



# Q4: Data Storage Model of Force.com
## Pivot tables

- Fallback Index Table
  - This index table is used in case the external search engine fails or is overloaded;
  - Stores the names of all the objects to enable a “fall-back” search operation to be performed.

- History Tracking Table
  - This table stores information about changes (field changes, object changes ..etc) carried out by a given user (indexed by OrgID) to enable tracking;
  - Useful in carrying out audit trails.

# Q4: Data Storage Model of Force.com Partitioning of Data and Metadata

- All the data structures (metadata, pivot table structures and data tables) used in the Force.com platform are partitioned by OrgID using native database partitioning mechanism

- Using the multi-tenant query optimizer, a given query from a given user will only be directed at the corresponding partitioned data structures.