**Force.com Database**

Database technology is the persistence layer at the heart of all data-centric applications, in change of organizing, protecting, and managing shared database access reliably, securely, efficiently.

The force.com database powers all of salesforce products today, serving more then 100,000 organizations, 135,000 applications, 3 million users, 1 billion transaction per day with an average request response time less than 300ms, all with an average up time of 99.9 + percent.

**Easy to Use**

With force.com, there is nothing to manage – the platform takes care of everything for you. Theirs is no software to install, update and patch. No worries about database back up and disaster recovery.

**Trustworthy**

Force.com is built with the security and privacy of customer information in mind, and meets all of the highest industry standards, including SAS 70 Type II, SysTrust and ISO 27001 certifications.

**Open**

Force.com’s full complement of open API’s lets you build and integrate applications using the approach of your choice, REST and SOAP based API’s are standards-based API’s that make force.com open to whatever programing language you want to use.

**Over View of Database Concepts**

In salesforce.com Database is an organized collection of objects. The term object refers to information. We use database tables to collect (or store) the information about things, people, or concept which are important for us and may require in the project for the further use. In salesforce we refer this information as object. In a database the information in the table is presented with rows and columns. Salesforce it’s maintained as record and field.

**Database Structure**

**Standard Object:** These are the objects that are pre-defined by the salesforce and readily available.

**Custom Objects:** These are the objects created by user according to their need. Each custom object has five standards fields

**External Objects:** These are the custom objects which are used to map the data stored outside your organization.

Apart from this salesforce.com provides some objects available in the API, we can access only authorized objects. The access to objects is determined by many factors such as object configuration, user permissions, access settings, data sharing model, and other factors related specifically to the object. Most of the objects accessible through the API have read-write access However few object are read-only.

**Relationships defined in Salesforce.com**

There are several relationships which can be defined on databases. Relationships associate one object with other object.  Relationship is always defined on the child object. The child object has complete access of the parent. Based on the handling of data deletion, record, ownership, security different types of relationships are categorized in to following ways:

**1) Many to One (Many child objects but one Parent Object)**

E.g. many metro cities are associated to One Country.

This kind of relationship is represented in four different forms like

* Master-detail
* Lookup (Loosely Coupled Relationship)
* Self
* Hierarchical

**Master-Detail (1: n)** — (One Parent Object and Many Child Objects)

* The record of child object gets automatically deleted when we delete the master object.
* To create a child object parent object reference field is required.
* Child object does not have separate sharing but it derives from Parent object, the detail records inherits the sharing and security settings of respective master record.
* The Owner field is automatically set to the owner of its associated master record as owner field is not available at detail side (child object).
* The detail record must have the master detail relationship field on its page layout.
* Administrators select Allow reparenting option in the master-detail relationship definition to represent child records in master-detail relationship on custom objects.

Master-detail relationships can be defined on two custom objects. One can define this relationship between one custom object and one standard object but in such relationship the standard object cannot be on the detail side of a relationship with a custom object. The data of custom object is displayed on page layout.

**Note:**  master-detail relationship can’t be created on objects where the User or Lead objects are the master.

**Lookup (1: n)**— Lookup is also one-to-many relationship but in this relationship two objects has no effect on deletion or security.

* Child objects are independent
* Child object have separate setting
* If we delete parent object child object will remain in system.
* Child may or may not have parent.

**Self:** In this relationship the object is self-referred. E.g. Add on Card on your credit card  
**Hierarchical**: This is also (1: N) Lookup relationship but can only be defined on User object. In this we can use lookup field to associate one user with other user. User objects may not directly or indirectly refer to itself.

**2)   Many-to-many**– master-detail relationships can be used to model many-to-many relationships between any two objects. In many-to-many relationship each record of one object is linked with multiple records from another objects and vice versa.  We need to create custom junction object to create a many-to-many relationship and then master-detail relationship fields, are linked with this objects.

* **Junction object:** This object is used to create Many-to-Many relationship. Salesforce support two relationship master detail and Lookup but both are 1:n that is one-to-many. To define many-to-many relationship we need third object and that is called junction object.

# **Salesforce Objects vs. SQL Server Database: Object Model vs. RDBMS Schema**

# **MS SQL Server DB vs. Salesforce Force.com Objects**

Objects are really much more than RDBMS tables. When you create an object – UI (page layout), search indexes and DR infrastructure are automatically generated for you. Bindings between your program and objects are automatically generated – unlike a RDBMS schema where you would need to create ADO.NET components (or prepared statements/JDBC in case of Java).

# **Data model in Force.com vs. RDBMS Schema**

You have a very rich set of data types (e.g. master-detail, lookups, picklists, formula) unlike a database table which has datatypes like char, varchar. The availability of standard objects and the rich data types mean that you can model very complex relationships with surprisingly few objects.

|  |  |
| --- | --- |
| RDBMS Tables | Objects |
| Tables | Objects |
| Columns | Fields |
| Keys (e.g primary key) | Ids, External Ids |
| Foreign Keys | Relationships |
| Keys (e.g primary key) | Ids, External Ids |
| Row | Record |