Domande e appunti

* **Che cos’è Salesforce?**

Salesforce è una web application che si usa tramite browser, su un server. È un CRM (Customer Relationship Management), piattaforma basata su cloud per costruire applicazioni. Salesforce offre servizi di infrastrutture, servizi applicativi e servizi di operazione.

Si aggiorna 3 volte l’anno. È “MULTI-TENANT CLOUD”, noi siamo “inquilini” del server cloud con un numero limitato di risorse, che però sono condivise con altri.

La divisione principale in Salesforce consiste in: “*Lightning Platform*”, ossia database, classi, oggetti, tutto ciò che mi serve per “costruire”, e “*Sales Cloud”*, che è il CRM di base, e il “*Service Cloud*”, che è invece un’estensione del CRM di base.

In aggiunta al CRM posso creare altre APP custom, come la gestione delle certificazioni, il tutto sempre partendo dalla lighting platform.

L’ultimo componente è CHATTER, una specie di forum dove è possibile condividere il lavoro (Social Enterprise Application o piattaforma di collaborazione).

Importantissimo in SF è la distinzione tra **Account** e **Contact**: Account è l’azienda (che può essere un cliente o un fornitore) che ci contatta, mentre il Contact è il dipendente di quell’account. Io devo gestire tutte le aziende che si abbonano verso Salesforce, i clienti e i fornitori.

* **Come si interagisce con Salesforce?**

L’utente interagisce, se si collega da pc e tablet, tramite browser, se da cellulare, con la app, attraverso l’API.

Livelli di un’app di SF: User Interface (view), Business Logic (i programmi controller), Data (object, model, entities, ecc.)

Le applicazioni su Salesforce hanno queste caratteristiche: accesso sicuro, automazione della logica, estrapolazione e analisi dei dati (grafici, ecc..), Data Management (backup, import, export, ecc.)

Salesforce si divide ulteriormente in “LOW CODE” e “PRO CODE”, anche se SF preferisce il Low code: per Low Code si intende il dichiarativo, ossia tutto ciò che ha a che fare con l’interfaccia, gli oggetti e i loro campi, le relazioni, model; per Pro Code si intende tutto ciò che ha a che fare con Apex, Flow, Visualforce, accesso agli oggetti ecc. Con il pro-code, inoltre, posso parlare con altri sistemi.

* **Cos’è la ORG?**

È l’insieme dei nostri dati e dei nostri programmi su Salesforce. È *data* e *metadata.* Dati, funzioni di base e automatismi.

METADATA: (flow, profili, permessi, OWD, ruoli, struttura degli oggetti e delle tabelle) è tutto il lavoro che facciamo su Salesforce che NON sono i valori dei campi, ma che possono essere inseriti da un utente mentre crea un record.

DATA: stato dei record, il contenuto dei metadata.

* **Cos’è un oggetto in Salesforce?**

È una tabella collegata a una classe, a un’interfaccia grafica, ai suoi programmi e alle sue relazioni. Gli oggetti standard sono già presenti in Salesforce, OOTB (out of the box), mentre quelli custom vengono creati da noi e siamo noi i “padroni”, e vengono riconosciuti dalla presenza di “\_\_c” alla fine.

Un oggetto in Salesforce ha dei campi “**TIPIZZATI**” già preesistenti. Quelli standard sono: Name, LastModifiedBy, ID, OwnerID. Gli altri campi vengono creati da noi e hanno l’\_\_c, perché sono *custom*.

Su Lightning platform ci sono anche i “System Object”, ossia oggetti per il funzionamento interno di Salesforce, che non possono essere modificati.

**SALESFORCE CONNECT**

Su SF ci sono anche i cosiddetti “External Objects”, che sono simili a oggetti custom, ma si riferiscono a dati salvati esternamente a Salesforce, accessibili tramite Salesforce Connect (che è uno strumento di integrazione di dati che connette dati esterni e fornisce loro le abilità tipiche della piattaforma base di SF al suo interno).

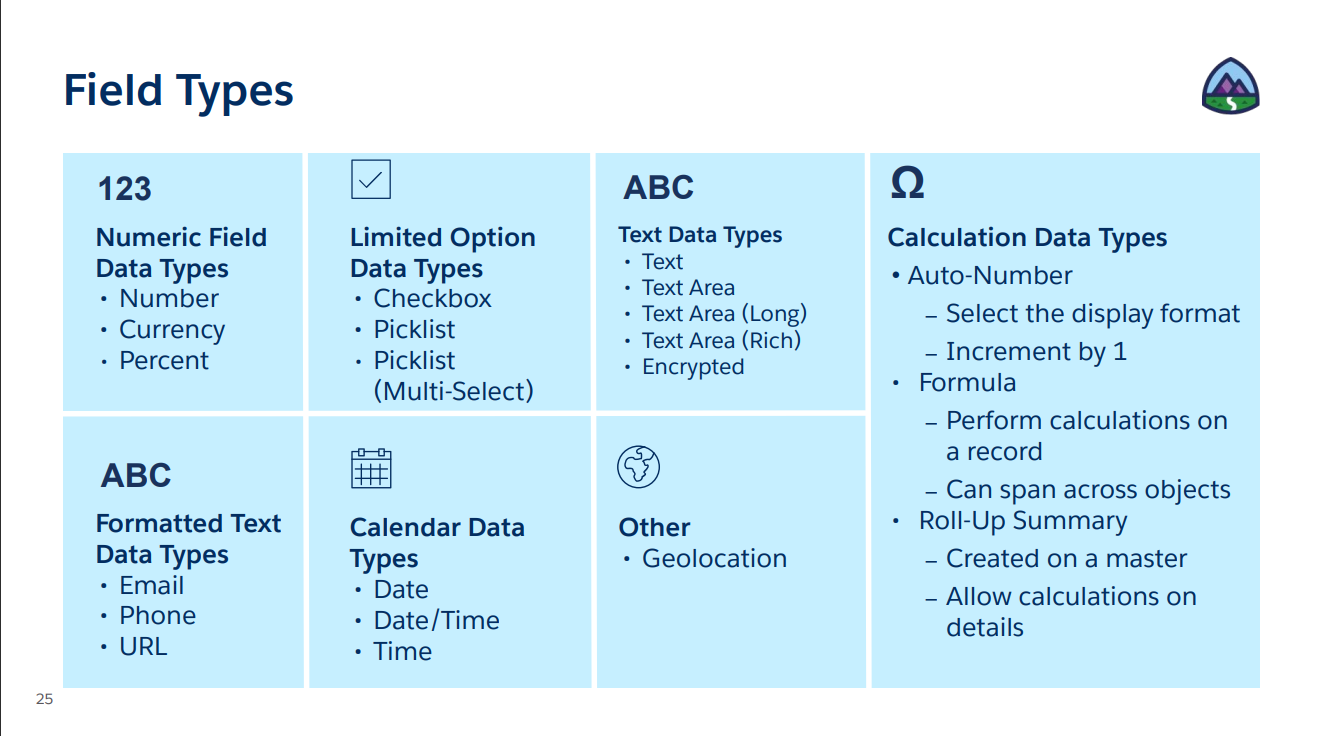
I dati, con SFConnect, sono accessibili in tempo reale, sono sempre aggiornati, puoi creare liste collegate ai dati esterni e questi possono essere collegati a oggetti SF Standard e Custom, come anche ad altri oggetti esterni.

* **Campi e relazioni in Salesforce**

Gli oggetti sono caratterizzati principalmente da campi e relazioni.

**CAMPI:**

Un campo può essere una proprietà o un metodo. Il “field label” e il “field name” sono il nome del campo che andiamo a creare. Il field label è il nome che viene mostrato all’utente, mentre il field name è il nome effettivo del campo, mostrato solo allo sviluppatore. Quando il campo viene creato, automaticamente viene generato l’API name, ossia il nome del campo usato nel codice.



**RELAZIONI:**

Esistono 2 tipi di relazioni in Salesforce, ossia **Master-Detail** e ­**Lookup Relationship**.

Il numero massimo di relazioni in un oggetto è 40 tra M-D e L-U, dove le M-D possono essere un massimo di 2 a oggetto, le lookup massimo 40. Se sono presenti le M-D, le lookup massime sono 38.

Le relazioni tra oggetti si basano su un “rapporto” padre-figli, in cui è presente un solo padre e tanti figli, in un rapporto 1-n.

Differenze: nelle **Master-Detail** il padre è sempre required nel figlio, mentre nelle **lookup** non è un campo richiesto di default, ma può essere opzionale (nelle M-D se il padre viene eliminato, “muoiono” anche i figli, mentre nelle lookup non è sempre detto a meno che non ci sia l’opzione “Don't allow deletion of the lookup record that's part of a lookup relationship.” ), nelle M-D l’accesso al padre determina l’accesso ai figli, mentre nelle lookup non c’è impatto a livello di sicurezza e accesso, l’accesso è indipendente al padre e ai figli. Nelle M-D il campo “rollup summary field” è disponibile (solo sul padre), mentre non è disponibile nelle lookup.

**UN OGGETTO STANDARD NON PUO’ ESSERE UN DETAIL**.

Nelle Master-Detail il figlio è sempre obbligato ad avere un riferimento al padre e può essere “rimparentato”, ossia il master può cambiare.

L’oggetto Detail non ha un campo OWNER perché è proprietà del Master.

Il Detail non può avere SHARING RULES.

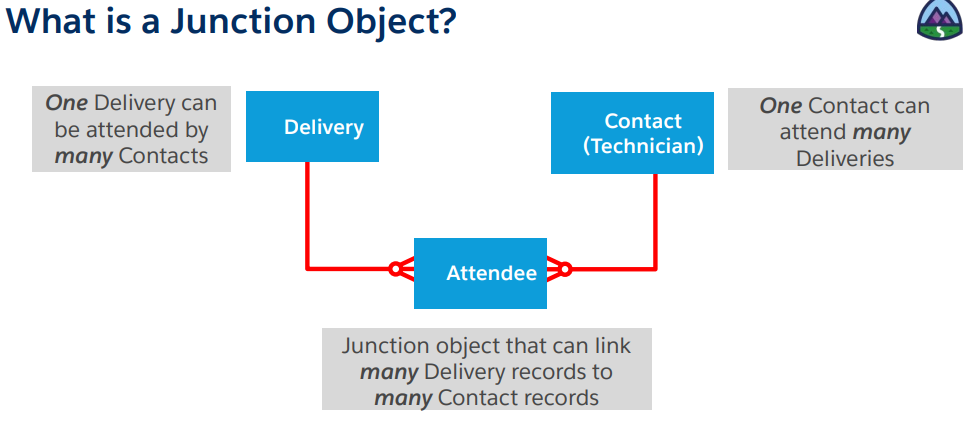
Un altro tipo di relazione, meno importante, è la “**self Relationship**”, ossia la relazione di un oggetto “con sé stesso”. In una Self Relationship gli oggetti vengono collegati allo stesso tipo (es: contact e contact), ma non a loro stessi. In una self Relationship collego un oggetto a me stesso: “tante storie possono essere collegate ad altre storie, ma non a loro stesse.”

La **Hierarchy** è una relationship tra user e user, in cui c’è un “capo” e i suoi “subordinati”. È molto simile a una self Relationship, ma a differenza di questa non può essere circolare. La Hierarchy si può fare solo su ACCOUNT e USER: su account sono presenti ma non creabili, mentre su user sono presenti e creabili.

Un **CROSS OBJECT FORMULA FIELD** è un campo formula che permette di prendere dal padre (partendo dal figlio) il contenuto di informazioni di un determinato campo, fino a un massimo di dieci relazioni di parentela.

Una **ROLL UP SUMMARY FIELD** è un campo disponibile solo su un Master di una relazione M-D e può effettuare un massimo di quattro operazioni sui figli: count, sum, max, min. Viene ricalcolato ogni volta che il record detail viene salvato o cancellato. L’operazione “sum” si può fare su campi number, currency, percentage.

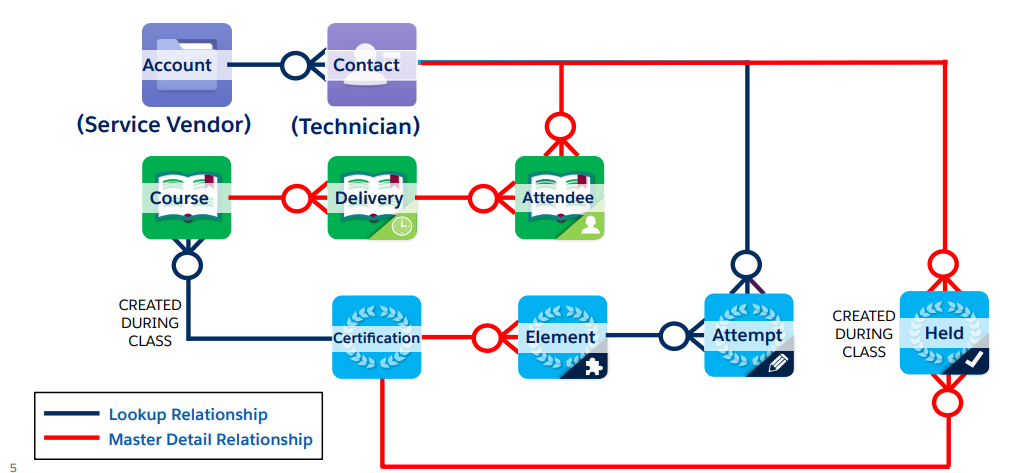
Un **JUNCTION OBJECT** è un oggetto custom che è dettaglio in due relazioni M-D (ha due Master). Permette di creare una relazione n-n tra due oggetti.



* **Cos’è un data model?**

È la struttura del nostro programma e del nostro database: mostra gli oggetti standard e le relazioni tra questi oggetti. Mostra che c’è un nucleo essenziale di componenti legati tra di loro da cui parte tutto.

È fondamentale per la comprensione della org, e si può vedere in forma riassuntiva nello Schema Builder dell’interfaccia di Setup.



L’oggetto “Attendee” è un Junction Object, perché è detail in due relazioni M-D.

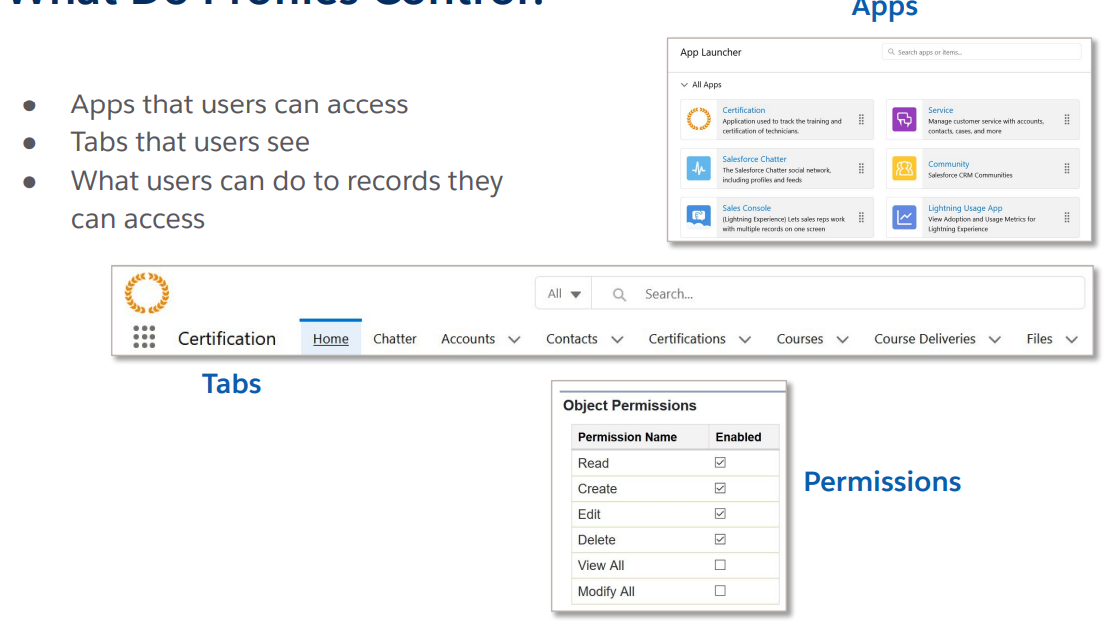
* **Cosa sono i PROFILI?**

Un profilo definisce i permessi di base per gli oggetti. Quando crei uno User devi assegnargli un profilo, che parte da un profilo standard. La relazione “profili-user” è una relazione 1-n: uno user può avere un solo profilo, un profilo può essere assegnato a tanti user.

Quando si crea un profilo, l’obiettivo è partire da un accesso minimo e aggiungervi permessi e accessi con vari “permission sets”.

Gli **ACCESSI** sono legati ai record, i **PERMESSI** sono legati agli oggetti.

**COSA CONTROLLA UN PROFILO?**





Le spunte view all e Modify All “Bucano lo sharing”, ossia permettono di vedere i record di quell’oggetto anche se privato e, nel caso di modify all, modificare a prescindere dalle restrizioni.

Le **system permission** le diamo per accedere a cose particolari, generalmente sono extra skills per profili particolari alcuni esempi:

- **Customize Application**: hai accesso al menu Setup, dato ai developer;

- **API ONLY USER**, non è un essere umano ma una macchina, quindi non ha permesso di entrare da browser;

- **Password never Expires**;

- **View All Data**: l’utente vede qualunque record di qualunque oggetto, indipendentemente dai suoi normali privilegi;

- **Modify All Data**: come sopra, ma può anche modificare tutto.

Per cambiare i permessi di un profilo a volte è meglio “rimparentare” l’utente, ossia ricreare il profilo e poi riassegnarlo, piuttosto che assegnargli troppi permission sets che andrebbero a collidere.

Nei profili viene definita anche la “**FIELD LEVEL SECURITY**”, ossia le restrizioni agli accessi ai campi per gli user, e viene “fornita” dal System Administrator o dall’owner dell’oggetto.

* **Cosa sono i PERMISSION SETS?**

I permission sets sono permessi aggiuntivi assegnati a uno specifico user in aggiunta ai permessi di base dati dal profilo. Uno user ha un solo profilo, ma può avere molteplici permission sets. I permessi dati dai permission sets vengono sempre sommati a quelli degli utenti, ma non rimuovono mai permessi già presenti.

I permission sets permettono a due utenti con lo stesso profilo, di avere permessi diversi sugli stessi oggetti.

I **PERMISSION SETS GROUPS** permettono di combinare molteplici permission sets in un gruppo di permission sets. Un permission set group modifica un permission set, rendendolo **mutato**. Un permission set mutato sarà tale solo all’interno del permission set group.

Anche se in un permission set group modifico, muto un permission set, non modifico i permessi di uno user.

* **Cos’è lo SHARING MODEL?**

Lo sharing model è l’insieme delle regole base di accesso ai record: determina a quali record ho accesso ed è determinato oggetto per oggetto.

Ci sono diversi tipi di Sharing Model:

1. WITH SHARING: onoro lo sharing model – accesso ridotto.
2. WITHOUT SHARING: “vedo tutto”. Non vi sono restrizioni (GOD MODE).
3. INHERITED SHARING: dipende da chi invoca i miei metodi. Se chi mi invoca ha with sharing, lo avrò anche io, altrimenti sarò without.
4. ANONYMOUS BLOCK (execute anonymous window nella developer console): rispetta lo sharing model e rispetta CRED e FLS.