

Phase 3: Data Modeling & Relationships

This phase focused on designing the **data structure** of the Insurance Claim & Policy Management System in Salesforce. The goal was to create **custom objects, fields, relationships, and page layouts** to capture all relevant data efficiently, ensure referential integrity, and support the business processes of policy management and claim tracking.

1. Standard & Custom Objects

Implementation:

- **Custom Objects Created:**
 - **Customer** – stores personal information of policyholders.
 - **Policy** – stores policy number, type, coverage amount, premium, start/end dates, and linked Customer.
 - **Claim** – tracks claim submissions, type, amount, status, and related Policy & Customer.
 - **Mentor/Agent** – stores agents assigned to customers.
- **Standard Objects Used:**
 - Account and Contact objects were minimally used; main focus was on custom objects.

Purpose:

- Maintain a modular, scalable structure.
- Each object represents a real-world entity, ensuring traceability and easy reporting.

Procedure:

1. Setup → **Object Manager** → Create → **Custom Object**.
 2. Enter Object Label, Plural Label, Record Name.
 3. Enable Allow Reports, Allow Activities, Track Field History.
 4. Save object and repeat for all required objects.
-

2. Fields

Implementation:

Object	Field Examples
Customer	Name, Email, Contact Number, Address
Policy	Policy Number, Type, Coverage Amount, Premium, Start Date, End Date
Claim	Claim Type, Claim Amount, Status, Submission Date

Object	Field Examples
Agent	Name, Email, Assigned Customers

Purpose:

- Capture all necessary information for claims, policies, and customer management.
- Field-level security applied according to user roles.

Procedure:

1. Object Manager → Select Object → **Fields & Relationships** → New.
 2. Choose field type (Text, Number, Date, Picklist).
 3. Set field-level security and add to **Page Layouts**.
-

3. Record Types

Implementation:

- Policy record types created for Health Insurance and Life Insurance.
- Different page layouts applied for each record type.

Purpose:

- Allows customized layouts and picklist values depending on policy type.

Procedure:

1. Object Manager → Policy → **Record Types** → New.
 2. Assign Page Layout and picklist values per record type.
 3. Activate and assign to profiles as needed.
-

4. Page Layouts & Compact Layouts

Implementation:

- Created custom page layouts for each object to show relevant fields to users.
- Added related lists for easy access to linked records (e.g., Customer → Policies → Claims).
- Compact layouts configured for key fields to show in record highlights.

Purpose:

- Improves UI/UX and ensures agents/managers see relevant information quickly.

Procedure:

1. Object Manager → Page Layouts → New / Edit Layout.

2. Drag fields, sections, and related lists.
 3. Assign layout to profiles/record types.
 4. Configure **Compact Layouts** → select key fields → assign to profiles.
-

5. Schema Builder

Implementation:

- Used **Schema Builder** to visually verify relationships between objects.
- Ensured lookup and master-detail relationships were correctly linked.

Purpose:

- Quick **visual validation** of object structure before testing automation.

Procedure:

1. Setup → **Schema Builder**.
 2. Drag objects to workspace → connect relationships.
 3. Save and validate field types and relationships.
-

6. Lookup vs Master-Detail Relationships

Implementation:

- **Customer → Policy:** Lookup Relationship (one customer can have multiple policies).
- **Policy → Claim:** Master-Detail Relationship (claim exists only if policy exists).
- **Agent → Customer:** Lookup Relationship.

Purpose:

- Ensures proper data linkage and referential integrity.
- Master-Detail relationship enables roll-up summary fields for analytics.

Procedure:

1. Object Manager → Field & Relationships → **New → Lookup / Master-Detail**.
2. Choose related object → define field-level security → add to page layouts.