Salesforce Admin Certification Study Guide

Object Manager and Lightning App Builder: 15%

■■ SALESFORCE OBJECT ARCHITECTURE

Understanding the Database Structure

Salesforce is fundamentally a cloud-based relational database where:

- Objects = Database tables
- Fields = Columns in tables
- Records = Rows in tables
- Relationships = Links between tables

Think of it like Excel: Objects are workbooks, fields are column headers, and records are individual rows of data.

Standard Objects

Definition: Pre-built objects that come out-of-the-box with Salesforce

Common Standard Objects:

- Accounts: Companies/organizations you do business with
- Contacts: People associated with accounts
- Leads: Potential customers not yet qualified
- Opportunities: Sales deals in progress
- Cases: Customer service inquiries/issues
- Campaigns: Marketing initiatives
- Products: Items/services your company sells
- Quotes: Proposed prices for products/services
- Contracts: Formal agreements with customers
- Tasks & Events: Activities and appointments

Key Characteristics:

- Cannot be deleted
- Come with pre-configured fields and relationships
- Standard fields cannot be deleted (but custom fields on standard objects can be)
- Can be customized by adding custom fields
- Already integrated with Salesforce features
- Have built-in page layouts and record types available

Exam Tip: You CANNOT upload custom fields on standard objects to AppExchange (only custom fields on custom objects can be packaged)

Custom Objects

Definition: Objects you create to store information specific to your business needs

When to Use Custom Objects:

- Standard objects don't fit your business process
- You need to track industry-specific data
- Your organization has unique requirements
- You want to extend Salesforce beyond CRM

Custom Object Characteristics:

- Always end with `__c` (e.g., `Property__c`, `Invoice__c`)
- Can be deleted (if not referenced elsewhere)
- Automatically include these standard fields:

- Identity Fields: Record ID (15 or 18 characters), Name
- System Fields: Created By, Last Modified By, Owner
- Name Field: Auto-generated field for record identification

- Start with zero custom fields (you add what you need)
- Can have up to 2 master-detail relationships
- Can have up to 40 lookup relationships (custom objects) or 25 (standard objects)

Creating Custom Objects:

1. Setup → Object Manager → Create → Custom Object
2. Define: Label, Plural Label, Object Name (API name)
3. Choose record name format: Text or Auto-Number

4. Enable features: Reports, Activities, Search, etc.

5. Create custom tab for UI access

Exam Scenario:

- Question: "Company needs to track solar panel installations with unique data"
- Answer: Create a custom object called "Installation" with relevant fields

---

■ OBJECT RELATIONSHIPS

Understanding Relationships

Relationships connect objects together, allowing data to flow between them. They're the foundation of Salesforce's relational database model.

---

1. Lookup Relationship

Nature: LOOSELY COUPLED - casual relationship

Key Characteristics:

- Optional by default (can be made required)
- Parent and child are independent
- Deleting parent does NOT delete child records
- Each object keeps its own sharing settings
- Child doesn't inherit parent's security
- No rollup summary fields allowed
- Can have up to 25 lookup relationships per object (standard) or 40 (custom)
- Represented by BLUE line in Schema Builder
- Can be left blank (null values allowed)
- One-to-many relationship (one parent, many children)

When to Use Lookup:

- Relationship is optional or reference-based
- Objects should remain independent
- Deleting parent shouldn't affect children

- Need flexibility in record ownership

Real-World Examples:

- Contact → Account (contacts can exist without accounts)
- Case → Contact (cases might not always have a related contact)
- Opportunity → Contact (optional relationship)
- Product → Vendor (products can exist independently)

Exam Key Point:

- Lookup = "Nice to have" relationship
- Child records survive parent deletion
- Each object has independent security

---

2. Master-Detail Relationship

Nature: TIGHTLY COUPLED - parent-child dependency

Key Characteristics:

- REQUIRED field on child (cannot be blank)
- Parent and child are dependent
- Cascade delete: Deleting parent DELETES all children
- Child inherits parent's sharing and security settings
- Child cannot have its own sharing rules
- Child record owner is automatically the parent record owner
- Enables rollup summary fields on parent
- Can have up to 2 master-detail relationships per object
- Represented by RED line in Schema Builder
- 10,000 detail records maximum per master record
- Detail record must always have a parent

Rollup Summary Fields (Master-Detail Only):

- COUNT: Number of child records
- SUM: Total of a number/currency field
- MIN: Minimum value of a field
- MAX: Maximum value of a field

When to Use Master-Detail:

- Child MUST have a parent
- Child should inherit parent's security
- Need automatic deletion of children
- Need aggregated data from children (rollup summaries)
- Tight data integrity required

Real-World Examples:

- Account → Opportunity (opportunities must belong to accounts)
- Opportunity → Opportunity Line Items (line items can't exist alone)
- Order → Order Line Items
- Invoice → Invoice Line Items
- Property → Offer (offers must be tied to properties)

Reparenting:

- Reparenting child records to different parent IS possible
- Must enable "Allow reparenting" option when creating relationship
- Useful when organizational structure changes

Exam Scenarios:

- "Delete property, all offers should also delete" → Master-Detail
- "Count total opportunities per account" → Master-Detail (for rollup)
- "Child should inherit parent security" → Master-Detail

---

3. Many-to-Many Relationship (Junction Objects)

Definition: Allows multiple records of one object to relate to multiple records of another object

How It Works:

- Created using a junction object (custom object in the middle)
- Junction object has TWO master-detail relationships
- First master-detail = Primary relationship (determines ownership)
- Second master-detail = Secondary relationship

Junction Object Characteristics:

- Custom object with two master-detail fields

- If EITHER parent is deleted, junction record is deleted
- Junction object name typically describes the relationship (e.g., "Student_Course__c")
- Can add custom fields to track relationship-specific data

Real-World Examples:

Example 1: Students & Courses

- Students can enroll in many courses
- Courses have many students
- Junction Object: "Enrollment"
- Fields on Enrollment: Student (MD), Course (MD), Grade, Semester

Example 2: Candidates & Job Positions

- Candidates apply to multiple positions
- Positions have multiple applicants
- Junction Object: "Job Application"
- Fields: Candidate (MD), Position (MD), Application Date, Status

Example 3: Projects & Consultants

- Consultants work on multiple projects
- Projects have multiple consultants
- Junction Object: "Project Assignment"
- Fields: Project (MD), Consultant (MD), Role, Hours Worked

Security Inheritance:

- Junction object inherits security from PRIMARY master-detail (first one created)
- Users need read access to BOTH parents to see junction record

Exam Tips:

- Key phrase: "Many-to-many" = Junction Object
- Two master-detail relationships create many-to-many
- Cannot have more than 2 master-detail per object (junction objects use both)

Exam Scenario:

- "Speakers present at multiple sessions, sessions have multiple speakers"
- Answer: Create junction object "Session Speaker" with two master-details

---

4. Hierarchical Relationship

Definition: Special relationship that links a record to another record of the SAME object

Key Points:

- ONLY available on User object (standard behavior)
- Creates parent-child hierarchy within same object
- Used for reporting structures (e.g., Manager field on User)
- Lookup relationship to the same object
- Can be up to 5 levels deep

Use Cases:

- Employee → Manager (both are Users)
- User reporting structure
- Organizational hierarchy

Exam Tip:

- Only User object has hierarchical relationship out-of-box
- Can create self-lookups on other objects but not called "hierarchical"

---

5. External Lookup Relationship

Definition: Links a Salesforce object to an external object (data stored outside Salesforce)

Use Case: Integration with external systems

Example: Link Salesforce Accounts to external billing system

Relationship Comparison Table

Feature	Lookup	Master-Detail	Many-to-Many
Coupling	Loose	Tight	Tight (via junction)
Required	Optional (can be required)	Always required	N/A (junction handles)
Cascade Delete	Optional	Yes (automatic)	Yes (if parent deleted)

| Security Inheritance | No | Yes | Yes (from primary MD) |

| Rollup Summaries | No | Yes | No |

| Max per Object | 40 (custom), 25 (standard) | 2 | N/A (uses 2 MDs) |

| Schema Builder Color | Blue line | Red line | Two red lines |

| Child Owner | Independent | Same as parent | Same as primary parent |

| Sharing Rules | Independent | Inherited from parent | Inherited from primary |

■ FIELDS

Field Types Overview

Salesforce offers 20+ field types. Here are the most important for the exam:

Text & String Fields:

- Text: Single-line (up to 255 characters)
- Text Area: Multi-line (up to 255 characters)
- Text Area (Long): Up to 131,072 characters
- Text Area (Rich): Formatted text with HTML
- Email: Validates email format
- Phone: Phone number
- URL: Website address

Number Fields:

- Number: Numerical values with decimals
- Currency: Money values (uses org currency or multi-currency)
- Percent: Percentage values
- Auto-Number: System-generated sequential number

Date & Time:

- Date: Calendar date only
- Date/Time: Date and timestamp
- Time: Time of day (no specific date)

Selection Fields:

- Picklist: Single-select dropdown (up to 1,000 values)

- Multi-Select Picklist: Multiple values (semicolon-separated)
- Checkbox: Boolean (true/false, yes/no)

Relationship Fields:

- Lookup Relationship: Links to another object (blue)
- Master-Detail Relationship: Parent-child link (red)
- External Lookup: Links to external object

Special Fields:

- Formula: Calculated field (read-only, derived from other fields)
- Rollup Summary: Aggregates child data (COUNT, SUM, MIN, MAX) - Master-Detail only
- Geolocation: Stores latitude/longitude
- Encrypted Text: Encrypted data at rest

Creating Fields

Steps:

1. Object Manager → Select Object → Fields & Relationships → New
2. Select field type
3. Enter field details (Label, Name, Length, etc.)
4. Set field-level security (which profiles can see/edit)
5. Add to page layouts

Important Field Properties:

- Required: Must have value to save (database-level)
- Unique: No duplicate values allowed
- External ID: Used for integrations (upsert operations)
- Default Value: Pre-populated value
- Help Text: Tooltip for users

Formula Fields:

- Read-only (calculated, not stored)
- Can reference fields from parent objects (cross-object formulas)
- Can return: Text, Number, Currency, Date, DateTime, Checkbox, Percent
- Used for calculations, concatenations, conditional logic

- Cannot be used in rollup summaries

Exam Tip:

- Formula fields are always READ-ONLY
- Rollup summaries only work with master-detail relationships

---

Deleting Fields - CRITICAL EXAM TOPIC

What Can Be Deleted:

- Custom fields on standard or custom objects: YES
- Standard fields on standard objects: NO

When You Delete a Field:

1. Data in the field is also deleted
2. Field is placed in Recycle Bin for 15 days
3. Field can be restored within 15 days
4. After 15 days, field is permanently deleted (cannot recover)
5. Deleted fields still count toward custom field limit until hard deleted

Implications of Deleting Fields:

1. Rollup Summary Fields Stop Calculating
   - If you delete a field used in a rollup summary formula
   - The rollup summary breaks and stops updating
   - Must update or delete the rollup summary field

2. Cross-Object Formulas Break
   - If formula field references a deleted field
   - Formula becomes invalid and shows error
   - Must update all formula fields referencing it

3. Picklist Dependencies Affected
   - If controlling picklist is deleted, dependent picklist loses dependency
   - Dependent values become inactive or available to all

4. Lookup/Master-Detail Relationships Impacted
   - Deleting relationship field removes connection between objects

- For master-detail: Child records might be deleted (if not reparented)
- Related lists disappear from parent records

5. Page Layouts
- Field remains on page layout as "unavailable"
- Must manually remove from layouts

6. Reports and Dashboards
- Reports using the field break
- Custom report types become invalid
- Dashboards stop refreshing

7. Validation Rules
- Rules referencing deleted field become invalid
- Must update or deactivate

8. Workflow Rules & Process Builder
- Automation using the field fails
- Processes become invalid

9. Apex Code & Triggers
- Code referencing field throws errors
- Must update code

10. Integrations
- API calls requesting field fail
- Integration mappings break

Restoring Deleted Fields:
- Restore within 15 days from Setup → Deleted Objects → Deleted Custom Fields
- Upon restoration:

- Field is NOT automatically added back to page layouts (must do manually)
- Required/Unique settings are NOT restored (must reconfigure)
- Field history tracking available again
- Master-Detail relationships restored as Lookup (must convert back)

Hard Delete (Permanent):
- If org is >75% of custom field limit, "Purge" button appears

- Purge permanently deletes fields from Recycle Bin
- Cannot be undone

Best Practices Before Deleting:

1. Check "Where is this used?" button (custom fields only)
2. Search in Dependency API results
3. Document field purpose in Description field
4. Backup data (Data Export or Data Loader)
5. Notify stakeholders
6. Check automation (Flows, Process Builder, Workflows)
7. Review reports and dashboards
8. Check Apex code
9. Consider hiding instead of deleting (remove from layouts, FLS)

Exam Scenarios:

- "Admin deletes field used in rollup summary" → Rollup stops calculating
- "Admin wants to restore deleted field after 20 days" → Cannot restore (permanently deleted after 15 days)
- "Master-Detail field deleted and restored" → Becomes Lookup (must convert back)

Changing Field Types

General Rules:

- Only custom fields can be changed (standard fields cannot)
- Some changes cause data loss
- Always backup data before changing field types

Cannot Change These Field Types:

- Auto-Number
- Formula
- Rollup Summary
- Geolocation
- External Lookup
- Master-Detail (to anything except Lookup)

Common Changes & Data Loss:

| From | To | Data Loss? |

|------|-----|-----------|

| Text | Picklist | No (if values match) |

| Picklist | Text | No |

| Number | Currency | No |

| Currency | Number | No |

| Text | Number | Yes (non-numeric values lost) |

| Checkbox | Any other | Yes (converts to Yes/No text) |

| Multi-Select Picklist | Picklist | Yes (multiple values consolidated) |

| Master-Detail | Lookup | No (but rollups deleted) |

| Lookup | Master-Detail | Possible (if no child orphans) |

Exam Tip:

- Know that changing field types can cause data loss
- Master-Detail → Lookup loses rollup summaries
- Backup before changing

Field Dependencies (Controlling vs. Dependent)

Definition: Filter picklist values based on another field's value

Controlling Field:

- Determines what's available in dependent field
- Can be: Checkbox or Picklist (standard or custom)
- Multi-Select Picklist CANNOT be controlling field

Dependent Field:

- Must be: Picklist or Multi-Select Picklist
- Values change based on controlling field selection

Example:

- Controlling Field: Country (Picklist: USA, Canada, Mexico)
- Dependent Field: State (Picklist: changes based on country)
- If "USA" selected → Shows US states
- If "Canada" selected → Shows Canadian provinces

Setup:

1. Setup → Object Manager → Fields & Relationships
2. Click "Field Dependencies" button
3. Select Controlling and Dependent fields
4. Map which dependent values show for each controlling value

Exam Scenario:

- "Manufacturer field should control Model field values"
- Answer: Manufacturer = Controlling Picklist, Model = Dependent Picklist
- Wrong Answer: Manufacturer = Dependent (exam trick!)

■ PAGE LAYOUTS

What Are Page Layouts?

Definition: Control the layout and organization of fields, buttons, and related lists on record detail and edit pages

What Page Layouts Control:

- Field placement and sections
- Field properties: Visible, Read-Only, Required (UI-level only)
- Related lists and their order
- Related list columns
- Standard and custom buttons
- Quick actions
- Publisher actions (Chatter)
- Custom links
- Visualforce pages
- Report charts

What Page Layouts DO NOT Control:

- Object-level permissions (controlled by profiles)
- Field-level security (controlled by FLS settings)
- Record visibility (controlled by OWD/sharing)

Key Points:

- One page layout per profile per record type per object
- If no record types: one page layout per profile
- Default page layout assigned to all users initially
- Can assign same layout to multiple profiles

Page Layout vs. Field-Level Security

Critical Distinction for Exam:

Feature	Page Layout	Field-Level Security (FLS)
Scope	UI only (detail/edit pages)	EVERYWHERE (reports, API, list views)
Security	NOT secure	Secure (true restriction)
Required Fields	UI validation only	Can be enforced
Read-Only	UI only	Enforced everywhere
Overrides	FLS overrides page layout	FLS always wins

Hierarchy: FLS > Page Layout

Exam Scenarios:

- "FLS hides field, Page Layout shows field" → Field is HIDDEN (FLS wins)
- "FLS shows field, Page Layout hides field" → Field is HIDDEN (Page Layout wins for visibility)
- "FLS is read-only, Page Layout is editable" → Field is READ-ONLY (FLS wins)
- "How to ensure sensitive data is hidden everywhere?" → Use FLS (NOT page layout)

Making Fields Required:

- Page Layout "Required": Only required on UI (API can bypass)
- Field-Level "Required": Database-level (cannot bypass)
- Validation Rule: Custom logic for required fields (most flexible)

Creating and Assigning Page Layouts

Creating:

1. Setup → Object Manager → Select Object → Page Layouts

2. Click "New" or "Clone" existing layout

3. Drag fields into sections

4. Organize related lists

5. Set field properties (required, read-only)

6. Save

Assigning:

1. Setup → Object Manager → Select Object → Page Layouts

2. Click "Page Layout Assignment"

3. Assign layouts to profiles

4. If record types exist, assign by profile AND record type

Best Practices:

- Clone existing layouts to save time
- Use clear naming conventions (e.g., "Account Layout - Sales")
- Keep layouts clean and organized
- Group related fields in sections
- Use section headers effectively

■ RECORD TYPES

What Are Record Types?

Definition: Enable different business processes, picklist values, and page layouts for different groups of users

What Record Types Control:

1. Business Processes (Sales Process, Support Process, Lead Process)

2. Page Layout Assignments (different layouts per record type)

3. Picklist Values (different values available per record type)

What Record Types DO NOT Control:

- Field-level security (use FLS)
- Object-level permissions (use profiles)
- Record visibility (use sharing)

Business Processes

Definition: Defines the stages/picklist values available for a specific process

Common Business Processes:

- Sales Process: Controls Opportunity Stage values
- Support Process: Controls Case Status values
- Lead Process: Controls Lead Status values
- Solution Process: Controls Solution Status values

Example:

- Sales Process 1 "Enterprise Sales": Stages = Prospecting, Qualification, Proposal, Negotiation, Closed Won, Closed Lost
- Sales Process 2 "SMB Sales": Stages = Discovery, Demo, Closed Won, Closed Lost
- Each sales process is tied to a record type

Key Points:

- One business process per record type
- Business process determines available picklist values
- Standard objects have pre-defined processes (Sales, Support, Lead)

Creating and Using Record Types

When to Use Record Types:

- Multiple business processes for same object
- Different picklist values needed
- Different page layouts needed based on process
- Different user experiences required

Creating Record Types:

1. Setup → Object Manager → Select Object → Record Types → New
2. Name the record type
3. Select business process (if applicable)
4. Choose which profiles have access

5. Set default record type per profile

6. Assign page layout for this record type

7. Configure picklist values

Assignment:

- Profile-level: Which record types can a profile access?
- Default record type: Which record type is pre-selected for profile?
- Page layout: Which layout shows for each profile + record type combo?

Picklist Value Assignment:

- Each record type can have different picklist values available
- Can reorder values per record type
- Can set default value per record type

User Experience:

- When creating new record, user sees "Record Type Selection" screen
- Can bypass selection screen if only one record type available
- Can set default record type to auto-select

Record Types vs. Page Layouts - Decision Matrix

Use Page Layout When:

- Only need to show/hide fields
- Same business process for all users
- Single view per user group (profile)

Use Record Type When:

- Multiple business processes needed
- Different picklist values required per process
- Need both different page layouts AND processes

Exam Examples:

Scenario A: "Support team needs to see different fields than Sales team on Account records"

- Answer: Create two page layouts (one for Support, one for Sales), assign to respective profiles
- DO NOT need record types (same business process)

Scenario B: "Enterprise sales have 7 opportunity stages, SMB sales have 4 stages, each needs different fields"

- Answer: Create 2 record types (Enterprise, SMB), 2 sales processes, 2 page layouts
- Need record types because different stages (business processes) required

Scenario C: "High priority cases need expedited approval, standard cases follow normal process"

- Answer: Create 2 record types (High Priority, Standard), 2 support processes, 2 page layouts
- Different processes = different record types needed

Scenario D: "Track residential and commercial properties with different fields for each"

- Answer: Create 2 record types (Residential, Commercial), 2 page layouts
- Even without business processes, record types useful for user experience

---

■ LIGHTNING APP BUILDER

What Is Lightning App Builder?

Definition: Drag-and-drop tool for creating custom Lightning pages without code

Page Types:

1. Home Pages: Customized home page for apps
2. Record Pages: Customized record detail pages
3. App Pages: Custom pages with tabs and components

Benefits:

- No coding required
- Responsive design (desktop and mobile)
- Reusable components
- Dynamic visibility rules
- Faster than Visualforce

Lightning Record Pages

What You Can Customize:

- Page structure and templates

- Component placement and configuration
- Tabs for fields and related lists
- Visibility of components (profile, record type, criteria-based)
- Mobile and desktop experiences

Page Templates:

- Header and One Column
- Header and Two Columns
- Header and Three Columns
- Header, Subheader, and Two Columns
- Header, Subheader, Right Sidebar
- Template CANNOT be changed after creation (must create new page)

Components:

Standard components available:

- Record Detail (fields)
- Related Lists
- Related Records
- Highlights Panel
- Chatter Feed
- Activities
- Path (visual process guide)
- Report Charts
- Rich Text
- Custom Lightning Components

Dynamic Forms:

- Allows individual field sections as components
- Fields can be dragged, hidden, reordered independently
- Better performance than traditional page layouts
- Not available on all objects (e.g., Campaigns)

Activating and Assigning Lightning Pages

Activation Options:

1. Org Default:

- Makes page the default for ALL users
- Simplest option but least flexible

2. App, Record Type, and Profile:

- Assign specific page to:

- Lightning App(s)

- Record Type(s)

- Profile(s)

- Most flexible option
- Can create different experiences per combination

3. Form Factor:

- Desktop: For users on computers
- Phone: For Salesforce mobile app users
- Can create separate pages for each

Assignment Priority (if multiple pages match):

1. App + Record Type + Profile

2. App + Record Type

3. App + Profile

4. Record Type + Profile

5. App Default

6. Org Default

Component Visibility Filters

Control when components show/hide based on:

1. Profile or Permission:

- Show component only to specific profiles
- Show if user has specific permission

2. Record Type:

- Show component only for specific record types

3. Field Value:

- Show if field meets criteria (e.g., Status = 'Closed')
- Can use multiple field conditions

4. Advanced Filter:

- Custom formula criteria
- Boolean logic (AND, OR)

Use Cases:

- Show "Renewal Reminder" only if contract expires in 30 days
- Show "Escalation" section only for High Priority cases
- Show "Partner Portal Link" only for Partner profiles
- Hide fields based on stage (e.g., hide pricing until Negotiation stage)

Exam Tip:

- Component visibility is MORE flexible than page layout visibility
- Can show/hide based on field values (page layouts cannot)

Lightning App Builder vs. Page Layouts

Key Differences:

Feature	Page Layout	Lightning App Builder
Field Control	Grouped in one block	Individual field sections (Dynamic Forms)
Structure	1-2 columns only	Up to 3 columns + sidebars
Components	Fields and related lists	Standard + custom components
Visibility Rules	Profile only	Profile, record type, field values
Tabs	No	Yes (can create tabs)
Rich Text	No	Yes
Report Charts	Limited	Yes (embed reports)
Mobile	Same as desktop	Can create separate mobile page

Which to Use:

Page Layouts:

- Simple field organization
- Quick customization
- Required for field-level properties (required, read-only)

Lightning App Builder:

- Complex page structures
- Custom components needed
- Dynamic visibility requirements
- Mobile-specific experiences
- Tabs and rich content

They Work Together:

- Lightning Record Page can contain "Record Detail" component
- Record Detail component gets fields from Page Layout
- Or upgrade to Dynamic Forms (fields become individual components)

■ SCHEMA BUILDER

What Is Schema Builder?

Definition: Visual tool to view, create, and modify objects, fields, and relationships

Key Features:

- Graphical representation of data model
- Drag-and-drop interface
- View relationships at a glance
- Create objects and fields visually
- Real-time updates

How to Use Schema Builder

Access:

1. Setup → Schema Builder
2. Select objects to view from left panel

3. Drag objects on canvas to arrange

Visual Elements:

Objects:

- Show as boxes on canvas
- List all fields inside
- Can drag to rearrange (visual only, doesn't change structure)

Relationships:

- Blue Lines: Lookup relationships
- Red Lines: Master-Detail relationships
- Crow's Feet (circle + arrow): One-to-many side (child object)
- Single Line: One side (parent object)

Fields:

- Listed inside object box
- Red vertical bar: Required field
- Field type icon shown
- Can view field properties by clicking

Buttons/Options:

- Auto-Layout: Automatically arranges objects
- Clear All: Removes all objects from canvas
- View Options: Display element names, hide relationships, hide legend
- Elements Tab: Create new objects, fields, relationships

Creating in Schema Builder

Create Objects:

1. Elements tab → Drag "Object" onto canvas
2. Fill in details (Label, Plural Label, API name)
3. Save

Create Fields:

1. Elements tab → Drag field type onto object

2. Choose from: Text, Number, Picklist, Lookup, Master-Detail, Formula, etc.

3. Fill in field details

4. Save

Create Relationships:

1. Drag "Lookup" or "Master-Detail" from Elements onto child object

2. Select parent object

3. Configure relationship settings

4. Save

Schema Builder Use Cases

1. Documentation:

- Show data model to stakeholders
- Onboard new team members
- Explain relationships visually

2. Planning:

- Design new features
- Map out data structure before building
- Identify gaps in data model

3. Troubleshooting:

- Trace data flow issues
- Find broken relationships
- Identify missing connections

4. Analysis:

- Understand complex orgs
- See object dependencies
- Evaluate relationship types

Exam Tip:

- Schema Builder is the BEST tool to view all objects and relationships visually
- Use Schema Builder to understand complex data models
- Cannot view validation rules, workflows, or processes in Schema Builder

---

■ EXAM STRATEGY & KEY CONCEPTS

Critical Distinctions

1. Lookup vs. Master-Detail

- Lookup: Loose (blue), optional, independent security, no rollups
- Master-Detail: Tight (red), required, inherited security, rollups allowed

2. Page Layout vs. Lightning Record Page

- Page Layout: Field organization, properties (required/read-only on UI)
- Lightning Page: Overall page structure, components, visibility rules

3. Field-Level Security vs. Page Layout

- FLS: Secure, enforced everywhere, overrides page layout
- Page Layout: UI only, not secure, for user experience

4. Record Types vs. Page Layouts

- Record Types: Different processes + picklists + layouts
- Page Layouts: Just field organization

5. Standard vs. Custom Objects

- Standard: Pre-built, cannot delete, integrated
- Custom: You create, can delete, industry-specific

Common Exam Scenarios

Scenario 1: "Students take many courses, courses have many students"

- Answer: Create junction object (e.g., "Enrollment") with two master-details

Scenario 2: "When property is deleted, all offers should be deleted"

- Answer: Master-Detail relationship (Property = master, Offer = detail)

Scenario 3: "Count number of opportunities per account"

- Answer: Rollup Summary field on Account (requires master-detail)

Scenario 4: "Admin deletes field used in formula"

- Answer: Formula