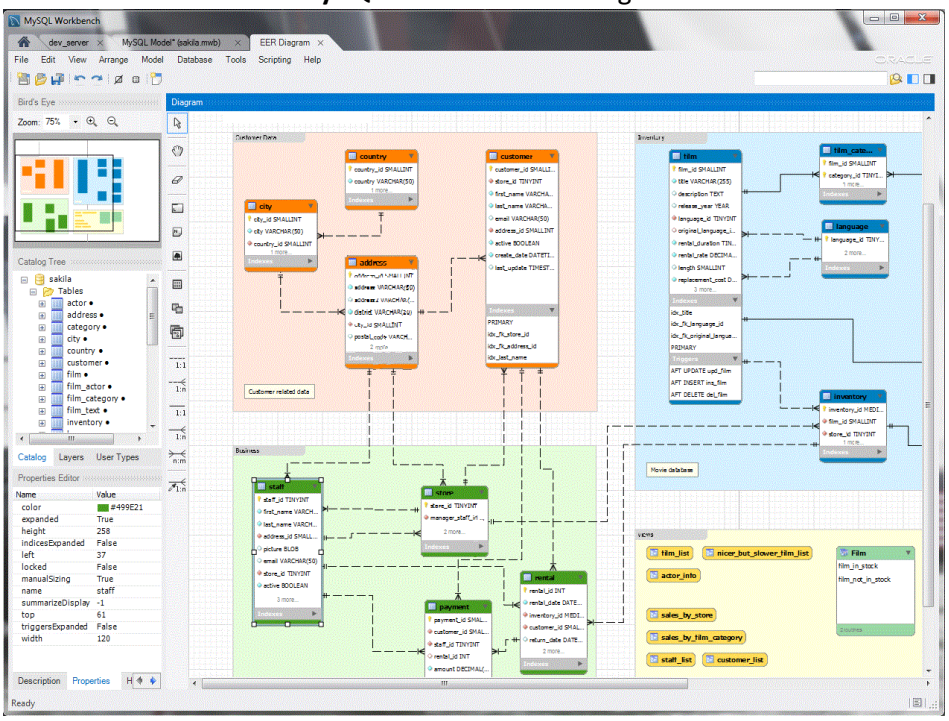
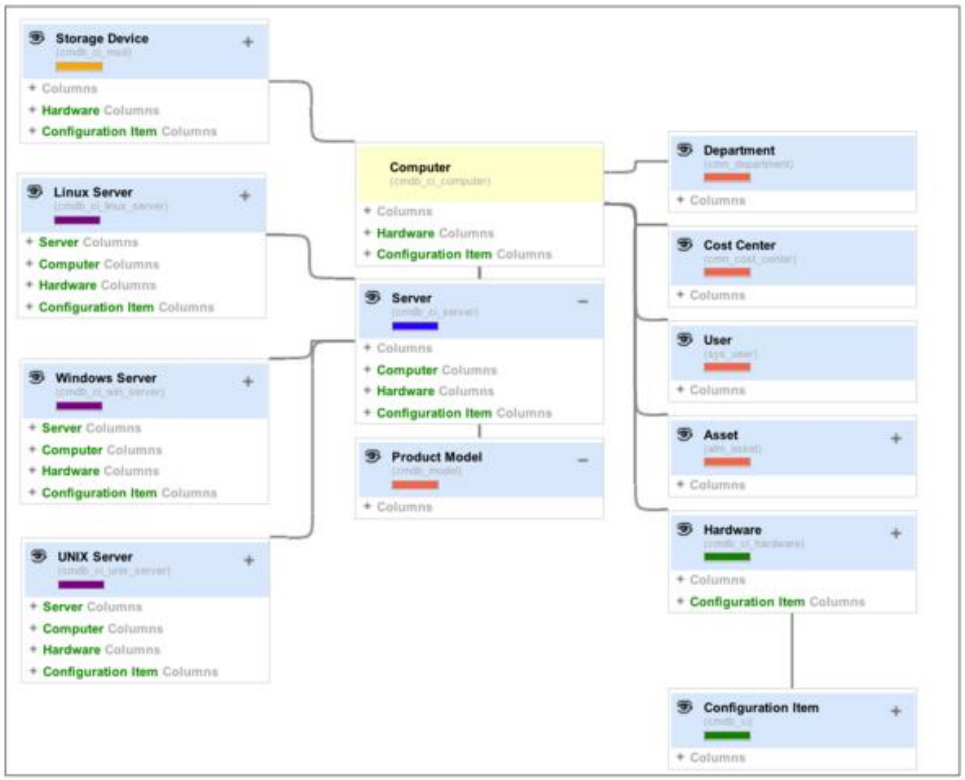
ServiceNow is built on a **MySQL** Database containing **tables**:



… which are accessible through the **ServiceNow GUI** (**Graphical User Interface**):



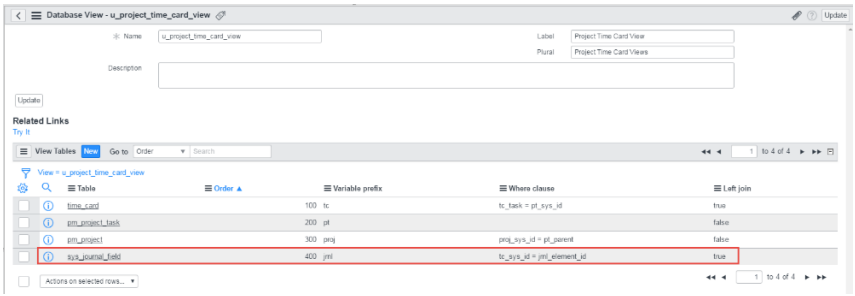
The structure of ServiceNow Data:

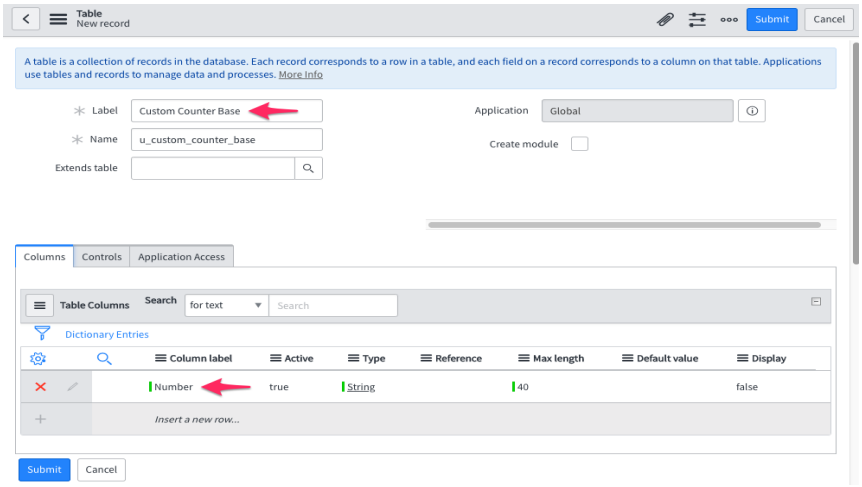
Database->

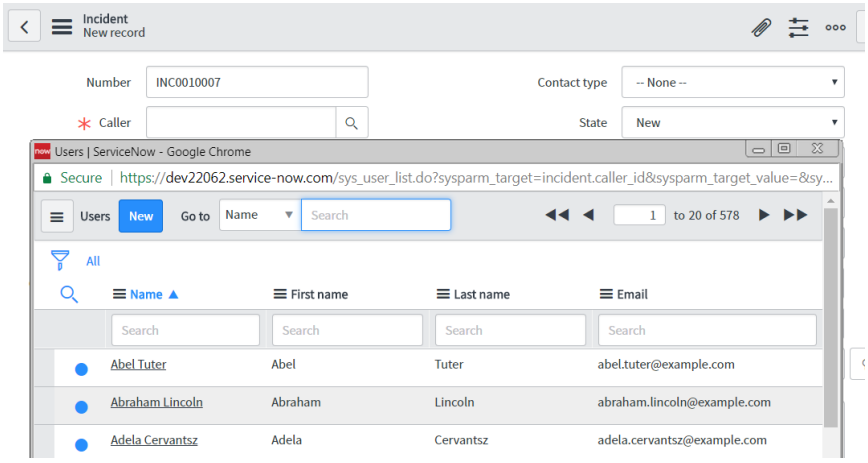
Tables->

Records->

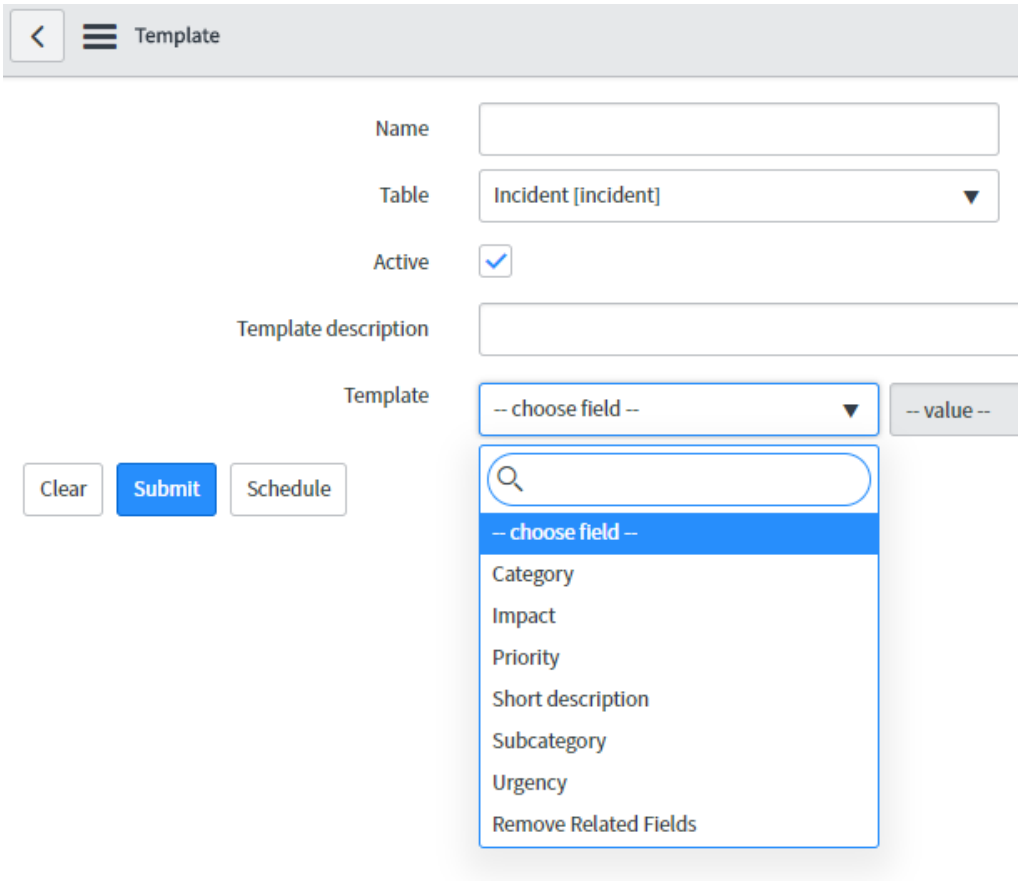
Fields





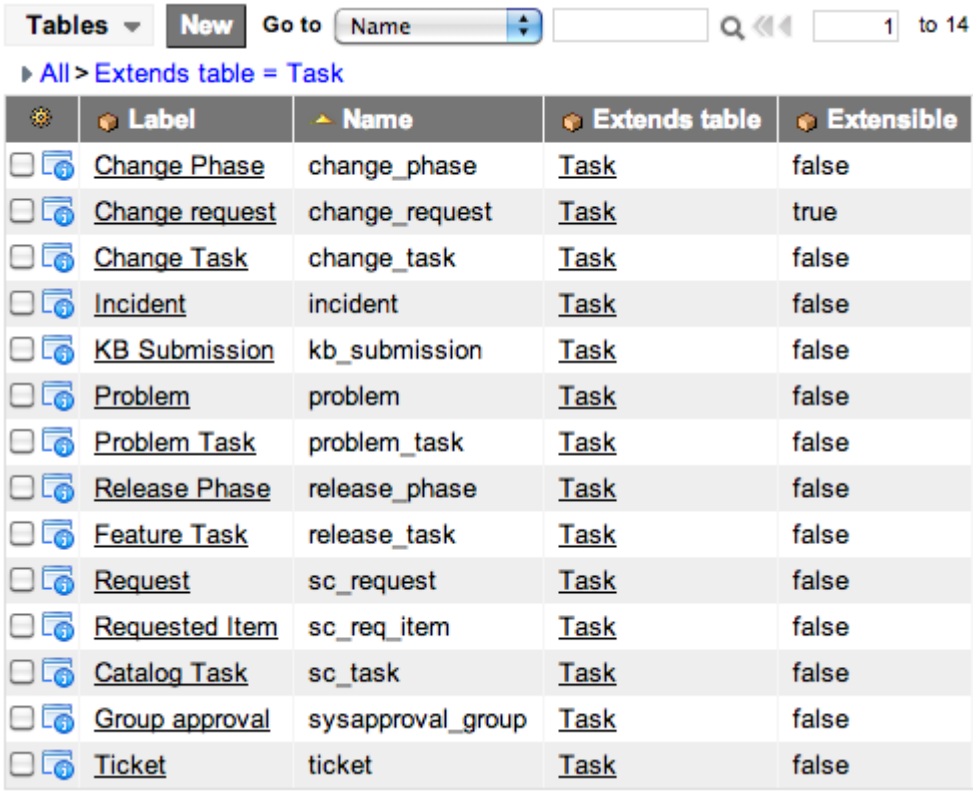


Data in ServiceNow is entered into individual **Fields**, which are displayed on a **Form**:



… and the term used to describe the data on a form is a **record**.

A ServiceNow table is made up of **records** and **fields**:



Tables | New | Go to Name | 1 to 14

All > Extends table = Task

Label	Name	Extends table	Extensible
Change Phase	change_phase	Task	false
Change request	change_request	Task	true
Change Task	change_task	Task	false
Incident	incident	Task	false
KB Submission	kb_submission	Task	false
Problem	problem	Task	false
Problem Task	problem_task	Task	false
Release Phase	release_phase	Task	false
Feature Task	release_task	Task	false
Request	sc_request	Task	false
Requested Item	sc_req_item	Task	false
Catalog Task	sc_task	Task	false
Group approval	sysapproval_group	Task	false
Ticket	ticket	Task	false

Each record is identified by a unique 32-character GUID (Globally Unique ID) called a Sys ID (**sys_id**). There are various ways to get the sys_id of a record:

Get the sys_id from the header bar

Users can locate the sys_id of a record using the header bar.

1. Navigate to the record.
2. Right click the header bar and select **Copy URL**.

   The sys_id is inside of the URL, after the parameter sys_id=. For example, the following is a URL for an Incident:

   ```
   https://<instance name>.service-now.com/nav_to.do?uri=incident.do?sys_id=9d385017c611228701d22104cc95c371
   ```

   Therefore the sys_id is 9d385017c611228701d22104cc95c371.

Get the sys_id from a script

Users can locate the sys_id of a record using a script.

- The sys_id value of a record can be found in a business rule (or any other server-side JavaScript) by dot-walking from the GlideRecord.

  ```
  var id = current.sys_id;
  ```

- The sys_id of a record can be found in client-side JavaScript using g_form.getUniqueValue() as shown in the following example.

  ```
  function onLoad() {
      var incSysid = g_form.getUniqueValue();
      alert(incSysid);
  }
  ```

Get the sys_id from the URL

Users can locate the sys_id of a record by viewing the URL.

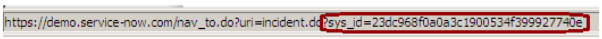
About this task

Since the sys_id of a record is always part of the URL for a link to that record, it is possible to retrieve the sys_id by viewing the URL.

Procedure

View the sys_id in the information bar of the browser by hovering over a link to the record.

For example, an Incident with the following URL: `https://<instance name>.service-now.com/nav_to.do?uri=incident.do?sys_id=23dc968f0a0a3c1900534f399927740e`, has this sys_id: 23dc968f0a0a3c1900534f399927740e.



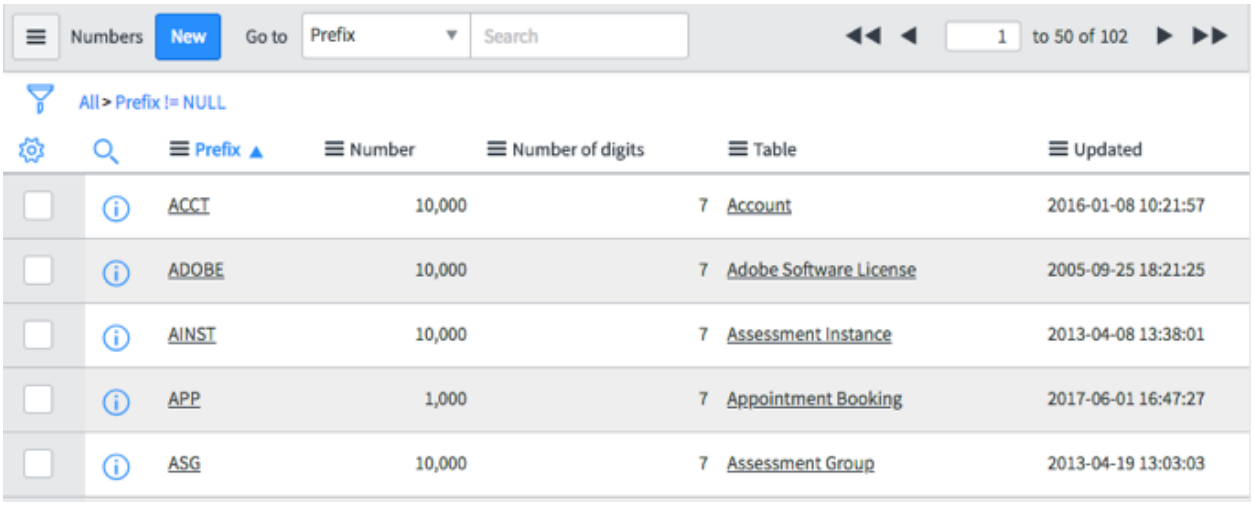
Which application is used to change the number format per table?

a. Number Maintenance
b. System Maintenance
c. Table Maintenance
d. Record Maintenance

A. Record numbers are automatically incremented, and the number format per table in the system can be changed by visiting the Number Maintenance application.

Number Maintenance: In the base system, several tables are numbered, including Incident, Problem, Change Request, and Knowledge. You can also use these numbers anywhere that script is present, for example to generate watermarks for emails. Records in tables can be numbered automatically.

Administrators can manage record numbering by navigating to **System Definition** > **Number Maintenance**. The current number format for a table, including the prefix (such as **INC** for incidents or **CHG** for changes), is stored in a record on the Number [sys_number] table.



The 3 components of a **Field**:

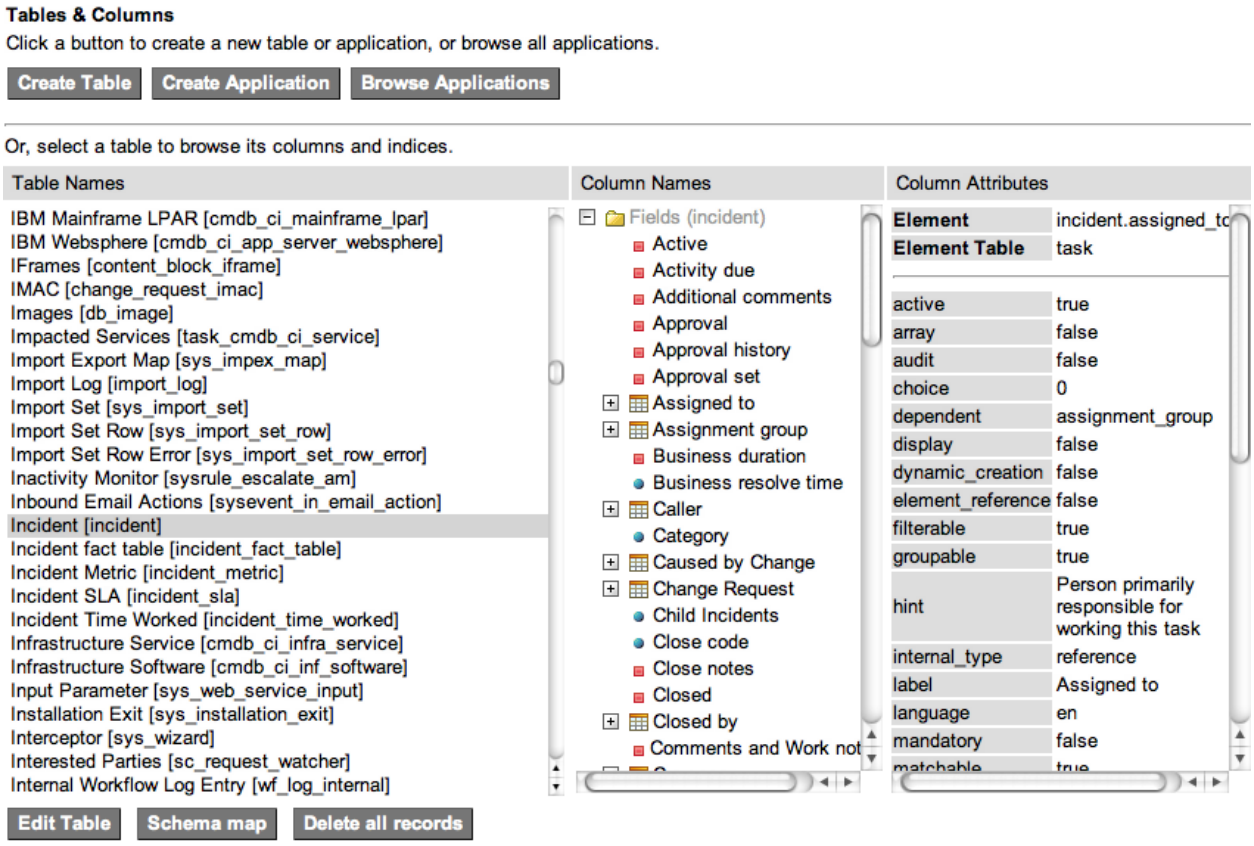
Label, Name, and Value.

Field label: is a user-friendly term which allows people to identify the field in the user interface.

Field name: The name is unique term that the system uses to identify the field in scripts and automated business processes.

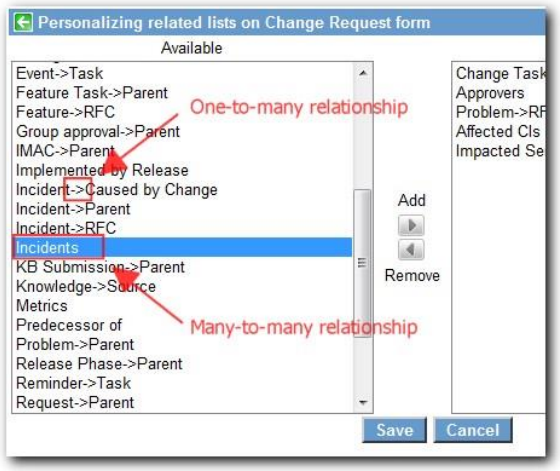
Value: The values are actual data

The **Tables & Columns** module provides a view-only list of all existing tables, with columns (fields), column (field) attributes, and indexes:

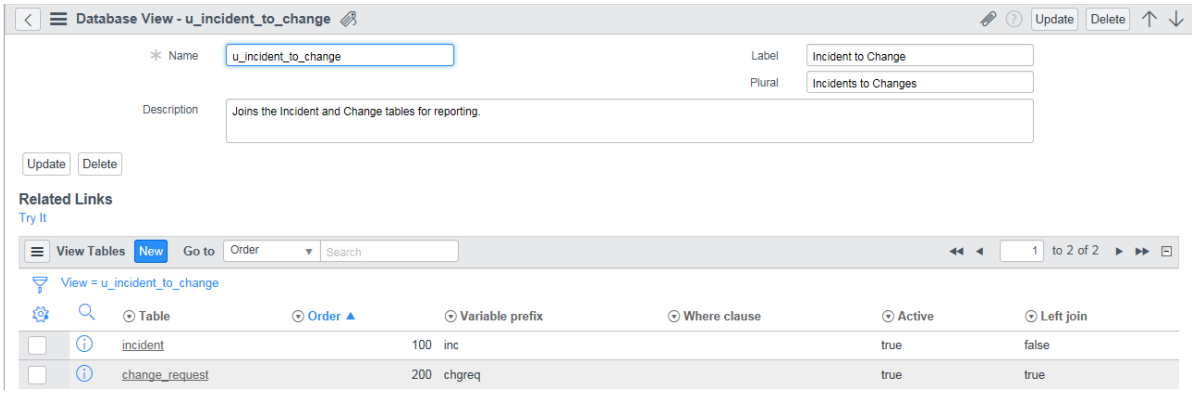


System Definition > Dictionary is where you can view a complete listing of records for all tables and table fields in the database.

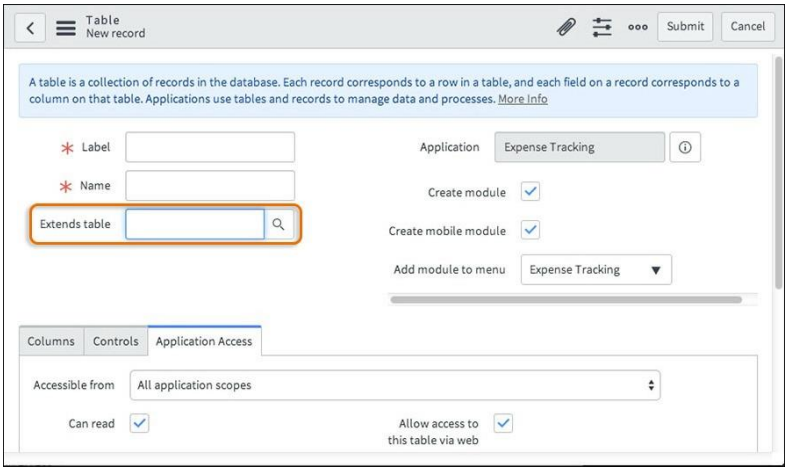
Relationships between Tables: The **One-to-many** and **Many-to-many** relationships:



The **Database View** between Incident Table and Change Request Table:



The Extension of a Table:



Within a table, a field can hold a reference to a record on another table. What are the three "one-to-many" relationship fields?

Reference Fields: Allows a user to select a record on a table defined by the reference field. Ex: Caller field on the incident table allows the user to select any record on User table.

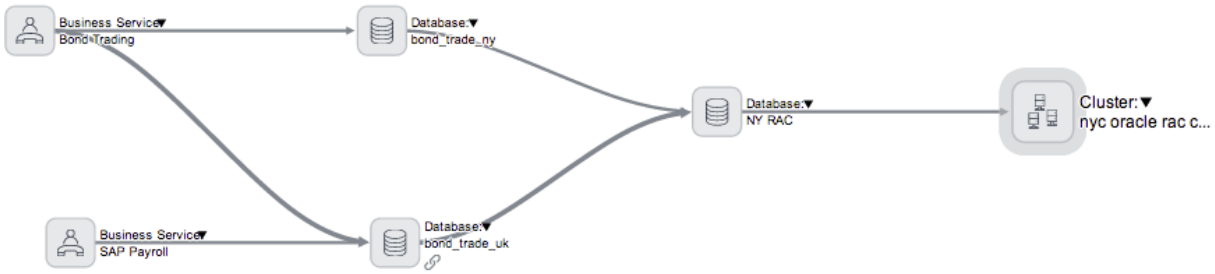
Glide List: Allows a user to select multiple records on a table defined by the glide list. Ex: The Watchlist field on the incident table allows the user to select any record or records on the User table.

Document ID Fields: Allows a user to select a record in the instance. Ex: Document field on the translated text table.

The **Dependency Views** Map display the CIs that support businesses, as well as the relationships between the CIs.

In a Dependency Views map, there is an indicator if a CI has any active, pending issues. You can investigate the tasks that are connected to a CI to get more details. When you return to the map from another form, the system restores the last map viewed, using the default filter and layout settings.

When you click the icon ( ) on a CI record or on a task record that identifies a CI, the map opens. Many of the relationships in map are created through the discovery process. You can also create, define, and delete CI relationships in the map. You can display the map from different perspectives and open specific records that relate to configuration items. The system refreshes the map automatically to reflect changes to the CMDB.



What are the four questions to ask when considering implementing a CMDB?

1. How is data entered or imported then managed?
2. Where is data stored?
3. What data is necessary?
4. When should imports or refreshes of CI data happen

What are some methods for populating the CMDB?

(1) Import Sets

(2) Integrate with External CMDB

(3) Manual Input

What is the purpose of an import set?

An Import set is a tool used to import data from various data sources, and map that data into ServiceNow tables.

What is the purpose of a transform map?

Provides a guide for moving data from Import Set tables to "Target" Tables; fields mapping provides direct field-to-field data moves.

What are some best practices for Importing data?

- Understand what data you are bringing in and where it should be placed.
- Plan time before an import to verify your data.
- Remove obsolete data before your data import.
- Inaccurate data takes time to fix after a data import.

What is a coalesce field?

In an import, coalescing on a field (or set of fields) means the field will be used as a unique key. If a match is found using the coalesce field(s), the existing record will be updated with the information being imported. If a match is not found, then a new record will be inserted into the database.

What are the three modules typically used by the system admin for security purposes?

- System properties > Security
- System Security > ACL
- System Security > High Security Settings

ServiceNow provides several levels of security before an end user can perform CRUD operations on a table. What are those three levels?

- User Authentication/ Login: Users, Groups, Roles
- Application and Modules Access: Controlled by roles configured at the application and module level.
- Database Access: Access to tables and their records and fields are controlled via globally defined system properties as well as table and field level access controls.

What are the 3 types of Access Control rules to identify the object being secured?

- Table.None: No specific field selected - this rule applies to the whole table including it's records.
- Table.field: This rules applies to only one field on a record.
- Table.*: This rule applies to every field on a record without a table.field rule.