¡Claro! Te entiendo perfectamente.  
Con todo lo que me enviaste, **te voy a hacer una guía completa, paso a paso, para crear secuencias en Oracle** basándome **solo** en esa teoría que me diste.

Te la organizo clara y práctica para que puedas seguirla como un procedimiento:

**📚 Guía Completa para Crear Secuencias en Oracle**

**1. ¿Qué es una Secuencia en Oracle?**

* Es un objeto que genera **números automáticos** de manera **ordenada**.
* Se usa para, por ejemplo, **asignar IDs únicos** a las filas de una tabla.

**2. Pasos para Crear una Secuencia**

**➡️ Paso 1: Escribir la instrucción básica**

CREATE SEQUENCE nombre\_de\_la\_secuencia;

👉 Esto crea una secuencia muy básica con valores predeterminados.

**➡️ Paso 2: Configurar Opciones (Opcionales pero Muy Importantes)**

Añade opciones para controlar **cómo funciona** la secuencia:

| **Opción** | **¿Qué hace?** | **Valor por defecto si no la pones** |
| --- | --- | --- |
| INCREMENT BY n | Número que suma o resta cada vez. | 1 |
| START WITH n | Número donde empieza. | Mínimo valor (ascendente) o máximo valor (descendente) |
| MAXVALUE n | Máximo número que puede generar. | 10^27 (ascendente) |
| MINVALUE n | Mínimo número que puede generar. | 1 (ascendente) |
| CYCLE | ¿Vuelve a empezar al llegar a límite? | No (por defecto es NOCYCLE) |
| CACHE n | Guarda números en memoria para más velocidad. | Sí (por defecto Oracle cachea) |
| ORDER | Garantiza el orden de entrega. | No ordena (NOORDER) |

**➡️ Paso 3: Ejemplo de secuencia configurada**

CREATE SEQUENCE id\_seq

INCREMENT BY 10

START WITH 10

MINVALUE 10

MAXVALUE 100

CYCLE

CACHE 2;

**¿Qué significa esto?**

* Empieza en 10.
* Suma 10 cada vez.
* Llega hasta 100 y luego vuelve a 10 (porque tiene CYCLE).
* Guarda 2 valores en memoria para ser más rápido.

**3. ¿Cómo Usar una Secuencia?**

**➡️ Obtener el siguiente valor:**

SELECT nombre\_secuencia.NEXTVAL FROM dual;

* Cada vez que llames NEXTVAL, te da un número nuevo.

**➡️ Obtener el valor actual:**

SELECT nombre\_secuencia.CURRVAL FROM dual;

* Te muestra el último número **que ya fue generado** en esta sesión.

**4. Usar Secuencias en Tablas**

**➡️ Antes de Oracle 12c (manualmente en INSERT)**

Primero creas una tabla:

CREATE TABLE tasks(

id NUMBER PRIMARY KEY,

title VARCHAR2(255) NOT NULL

);

Luego creas una secuencia:

CREATE SEQUENCE task\_id\_seq;

Cuando insertas datos, usas NEXTVAL:

INSERT INTO tasks(id, title) VALUES(task\_id\_seq.NEXTVAL, 'Primer Task');

INSERT INTO tasks(id, title) VALUES(task\_id\_seq.NEXTVAL, 'Segundo Task');

**➡️ Desde Oracle 12c (usando columnas IDENTITY)**

Oracle puede **generar los números automáticamente**, sin usar NEXTVAL.

Crear tabla con columna identity:

CREATE TABLE tasks(

id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

title VARCHAR2(255) NOT NULL

);

Y ahora puedes insertar sin mencionar el id:

INSERT INTO tasks(title) VALUES('Tarea nueva');

✅ Oracle se encarga solo de crear y usar la secuencia **internamente**.

**5. Detalles importantes**

| **Tema** | **Resumen** |
| --- | --- |
| **CYCLE** | Si lo pones, vuelve al inicio cuando alcanza máximo o mínimo. |
| **NOCYCLE** | Si no pones CYCLE, se detiene y da error al llegar al límite. |
| **CACHE** | Mejora velocidad, pero se pierden números si se cae el servidor. |
| **ORDER** | Asegura que los números se den en orden de solicitud, importante en RAC (Oracle en Clúster). |

**📝 Resumen rápido de pasos**

1. **CREATE SEQUENCE nombre**.
2. **Configura**: INCREMENT, START, MAXVALUE, MINVALUE, CYCLE, CACHE, ORDER.
3. **Usa NEXTVAL** para insertar en tablas.
4. **Si usas identity (desde Oracle 12c)**, Oracle lo hace automáticamente.

¿Te gustaría que también te prepare **un esquema visual tipo mapa mental** para esta guía, así te lo aprendes aún más rápido? 🚀  
¿Te lo hago? 🎨✨

¡Perfecto! Me alegra ver que tienes toda esta teoría organizada.  
Te hago un resumen breve para que tengas claro lo importante antes de que lo apliquemos en ejercicios:

**Columnas de identidad en Oracle 12c**

Antes de Oracle 12c, **no existía** una forma directa de hacer autonumeración. Se usaban **secuencias** + **triggers**.

Desde Oracle 12c (versión 12.1), existen **Columnas de Identidad** (IDENTITY) de forma nativa.

**Sintaxis Básica**

GENERATED [ ALWAYS | BY DEFAULT [ ON NULL ] ] AS IDENTITY [ (opciones) ]

* **ALWAYS**:
  + Siempre usa la secuencia interna.
  + Si intentas insertar un valor manualmente (aunque sea NULL), da error.
* **BY DEFAULT**:
  + Usa la secuencia **si no insertas ningún valor**.
  + Pero **puedes** insertar un valor manualmente.
* **BY DEFAULT ON NULL**:
  + Usa la secuencia **si insertas NULL**.
  + También puedes poner un valor manualmente.

**Ejemplos rápidos**

**ALWAYS**

CREATE TABLE tabla (

id NUMBER GENERATED ALWAYS AS IDENTITY,

nombre VARCHAR2(50)

);

* No puedes hacer INSERT INTO tabla (id, nombre) VALUES (5, 'Pepe'); (error).
* Solo puedes hacer INSERT INTO tabla (nombre) VALUES ('Pepe');

**BY DEFAULT**

CREATE TABLE tabla (

id NUMBER GENERATED BY DEFAULT AS IDENTITY,

nombre VARCHAR2(50)

);

* Puedes dejar que se genere solo o **puedes** darle un valor.

**BY DEFAULT ON NULL**

CREATE TABLE tabla (

id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,

nombre VARCHAR2(50)

);

* Si insertas NULL, **se genera automáticamente** el valor de la secuencia.
* Si insertas otro número, lo respeta.

**Otras cositas importantes**

* Solo **una columna de identidad** por tabla.
* Siempre son **NOT NULL**.
* Internamente Oracle crea una **secuencia** con nombre como ISEQ$$\_XXXXX.
* No hereda identidad si haces CREATE TABLE AS SELECT.

**Rendimiento**

* Usar columnas IDENTITY **es mucho más rápido** que usar un **trigger**.
* **Directo con secuencia** y **IDENTITY** tienen tiempos muy parecidos (y ambos mejores que el trigger).

¿Quieres que ahora pasemos a hacer unos ejercicios prácticos tipo examen con columnas de identidad? 🚀  
(Te puedo preparar 5 para empezar). ¿Te gustaría?