

Fair Use Notice:

The material used in this presentation i.e., pictures/graphs/text, etc. is solely intended for educational/teaching purpose, offered free of cost to the students for use under special circumstances of Online Education due to COVID-19 Lockdown situation and may include copyrighted material - the use of which may not have been specifically authorised by Copyright Owners. It's application constitutes Fair Use of any such copyrighted material as provided in globally accepted law of many countries. The contents of presentations are intended only for the attendees of the class being conducted by the presenter.

DATABASE SYSTEMS (SW215)

SEQUENCE

By : HIRA NOMAN

SEQUENCE

- Sequences are database objects from which multiple users can generate unique integers.
- The sequence generator generates sequential numbers, which can help to generate unique primary keys automatically.
- Without sequences, sequential values can only be produced programmatically.
- Sequences eliminate serialization and improve the concurrency of an application.
- Sequence numbers are generated independently of tables, so the same sequence can be used for one or for multiple tables.
- After a sequence is created, you can access its values in SQL statements with the **CURRVAL pseudo column**, which returns the current value of the sequence, or the **NEXTVAL pseudo column**, which increments the sequence and returns the new value.

- To create a sequence in your schema, you must have the **CREATE SEQUENCE** system privilege.
- To create a sequence in another user's schema, you must have the **CREATE ANY SEQUENCE** privilege.

SYNTAX

CREATE SEQUENCE schema_name.sequence_name

[**INCREMENT BY** interval]

[**START WITH** first_number]

[**MAXVALUE** max_value | **NOMAXVALUE**]

[**MINVALUE** min_value | **NOMINVALUE**]

[**CYCLE** | **NOCYCLE**]

[**CACHE** cache_size | **NOCACHE**]

[**ORDER** | **NOORDER**]

```
CREATE SEQUENCE schema_name.sequence_name  
[ INCREMENT BY interval ]
```

Schema_name

- Specify the schema to contain the sequence. If you omit schema, then Oracle Database creates the sequence in your own schema.

INCREMENT BY

- Specifies the interval between sequence numbers.
- This integer value can be any positive or negative integer, but it cannot be 0.
- This value can have 28 or fewer digits.
- If this value is negative, then the sequence descends.
- If the value is positive, then the sequence ascends.
- If you omit this clause, then the interval defaults to 1.

[**START WITH** first_number]

START WITH

- Specifies the first sequence number to be generated.
- Use this clause to start an ascending sequence at a value greater than its minimum or to start a descending sequence at a value less than its maximum.
- For ascending sequences, the default value is the minimum value of the sequence.
- For descending sequences, the default value is the maximum value of the sequence.
- This integer value can have 28 or fewer digits

[MAXVALUE max_value | NOMAXVALUE]
[MINVALUE min_value | NOMINVALUE]

MAXVALUE

- Specifies the maximum value the sequence can generate.
- This integer value can have 28 or fewer digits.
- MAXVALUE must be equal to or greater than START WITH and must be greater than MINVALUE.

NOMAXVALUE

- Specify NOMAXVALUE to indicate a maximum value of 10^{27} for an ascending sequence or -1 for a descending sequence.
- This is the default.

MINVALUE

- Specifies the minimum value of the sequence.
- This integer value can have 28 or fewer digits.
- MINVALUE must be less than or equal to START WITH and must be less than MAXVALUE.

NOMINVALUE

- Specify NOMINVALUE to indicate a minimum value of 1 for an ascending sequence or -10^{26} for a descending sequence.
- This is the default.

[CYCLE | NOCYCLE]
[CACHE cache_size | NOCACHE]
[ORDER | NOORDER]

CYCLE

- Specify CYCLE to indicate that the sequence continues to generate values after reaching either its maximum or minimum value.
- After an ascending sequence reaches its maximum value, it generates its minimum value.
- After a descending sequence reaches its minimum, it generates its maximum value.

NOCYCLE

- Specify NOCYCLE to indicate that the sequence cannot generate more values after reaching its maximum or minimum value.
- This is the default.

CACHE

- Specify how many values of the sequence the database pre allocates and keeps in memory for faster access.
- This integer value can have 28 or fewer digits. The minimum value for this parameter is 2.
- For sequences that cycle, this value must be less than the number of values in the cycle.
- You cannot cache more values than will fit in a given cycle of sequence numbers. Therefore, the maximum value allowed for CACHE must be less than the value determined by the following formula:

$(\text{CEIL}(\text{MAXVALUE} - \text{MINVALUE})) / \text{ABS}(\text{INCREMENT})$

- If a system failure occurs, then all cached sequence values that have not been used in committed DML statements are lost. The potential number of lost values is equal to the value of the CACHE parameter.

[**CACHE** cache_size | **NOCACHE**]
[**ORDER** | **NOORDER**]

- Oracle recommends using the CACHE setting to enhance performance.

NOCACHE

- Specify NOCACHE to indicate that values of the sequence are not pre allocated.
- If both CACHE and NOCACHE are omitted, then the database caches 20 sequence numbers by default.

ORDER

- Specify ORDER to guarantee that sequence numbers are generated in order of request.
- This clause is useful if you are using the sequence numbers as timestamps.
- Guaranteeing order is usually not important for sequences used to generate primary keys.

NOORDER

- Specify NOORDER if you do not want to guarantee sequence numbers are generated in order of request.
- This is the default.

EXAMPLE:

```
CREATE SEQUENCE id_seq2  
INCREMENT BY 10  
START WITH 10  
MINVALUE 10  
MAXVALUE 100  
CYCLE  
CACHE 2 ;
```

The above statement creates an ascending sequence called `id_seq2`, starting from 10, incrementing by 10, minimum value 10, maximum value 100. The sequence returns 10 once it reaches 100 because of the `CYCLE` option. Database pre allocates two values in the memory because of `CACHE` option.

```
CREATE SEQUENCE id_seq2  
INCREMENT BY 10  
START WITH 10  
MINVALUE 10  
MAXVALUE 100  
CYCLE  
CACHE 2 ;
```

```
SELECT id_seq2.NEXTVAL  
FROM dual  
CONNECT BY level <= 12;
```

Results Script Output Explain Autotr

Results:

	NEXTVAL
1	10
2	20
3	30
4	40
5	50
6	60
7	70
8	80
9	90
10	100
11	10
12	20

REFERENCING A SEQUENCE

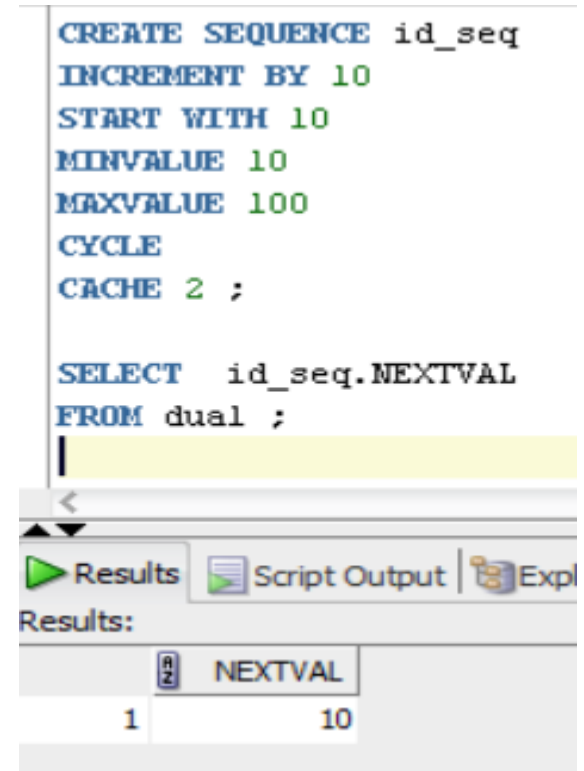
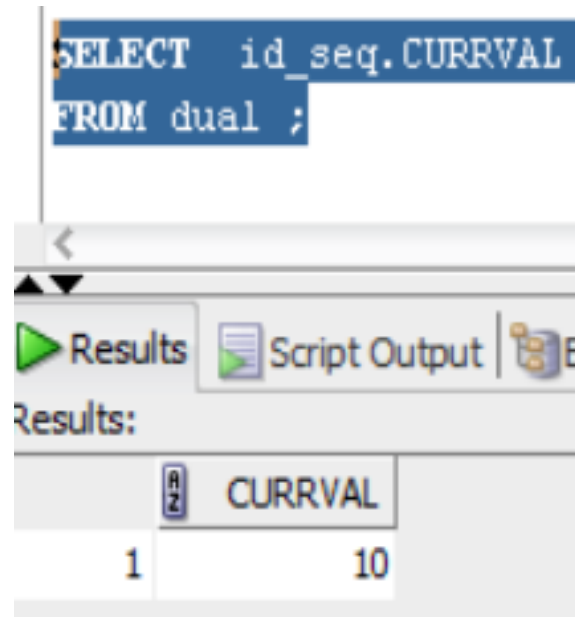
- A sequence is referenced in SQL statements with the NEXTVAL and CURRVAL pseudo columns; each new sequence number is generated by a reference to the sequence pseudo column NEXTVAL, while the current sequence number can be repeatedly referenced using the pseudo-column CURRVAL.
- NEXTVAL and CURRVAL are not reserved words or keywords and can be used as pseudo column names in SQL statements such as SELECT, INSERT, or UPDATE.

Using the NEXTVAL pseudo-column

```
SELECT id_seq.NEXTVAL  
FROM dual ;
```




Using the CURRVAL pseudo-column


```
SELECT id_seq.CURRVAL  
FROM dual ;
```






```
CREATE SEQUENCE id_seq  
INCREMENT BY 10  
START WITH 10  
MINVALUE 10  
MAXVALUE 100  
CYCLE  
CACHE 2 ;
```


```
SELECT id_seq.NEXTVAL  
FROM dual ;
```

Results:   




	 NEXTVAL
1	10


```
SELECT id_seq.CURRVAL  
FROM dual ;
```

Results:   

	 CURRVAL
1	10

```
SELECT id_seq.CURRVAL  
FROM dual ;  
SELECT id_seq.NEXTVAL  
FROM dual ;
```

Results:   

	 NEXTVAL
1	20

USING SEQUENCE NUMBERS WITH NEXTVAL

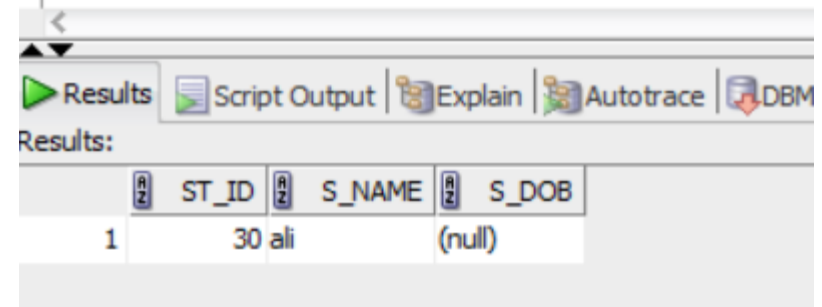
- To generate and use a sequence number, reference seq_name.NEXTVAL.

EXAMPLE

```
INSERT INTO sw_students2 (st_id, s_name)  
VALUES (Id_seq.NEXTVAL,'ali');
```

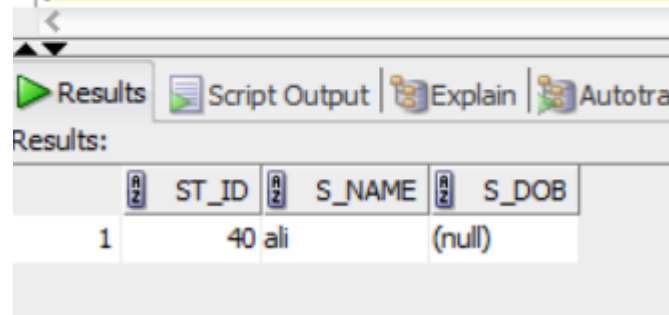
```
UPDATE sw_students2  
SET st_id = id_seq.NEXTVAL  
WHERE s_name = 'ali';
```

```
INSERT INTO sw_students2 (st_id, s_name)  
VALUES (Id_seq.NEXTVAL, 'ali');  
  
select * from sw_students2
```



	ST_ID	S_NAME	S_DOB
1	30	ali	(null)

```
UPDATE sw_students2  
SET st_id = id_seq.NEXTVAL  
WHERE s_name = 'ali';  
  
select * from sw_students2
```



	ST_ID	S_NAME	S_DOB
1	40	ali	(null)

USING SEQUENCE NUMBERS WITH CURRVAL

- To use or refer to the current sequence value ,reference seq_name.CURRVAL.
- CURRVAL can only be used if seq_name.NEXTVAL has been referenced in the current user session (in the current or a previous transaction).
- CURRVAL can be referenced as many times as necessary, including multiple times within the same statement.
- The next sequence number is not generated until NEXTVAL is referenced.

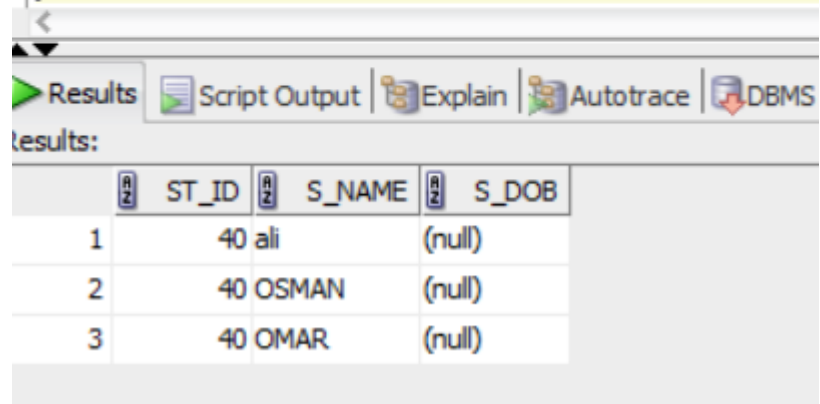
EXAMPLE

```
INSERT INTO sw_students2 (st_id, s_name)
VALUES (Id_seq.CURRVAL,'OSMAN');
```

```
INSERT INTO sw_students2 (st_id, s_name)
VALUES (Id_seq.CURRVAL,'OSMAN');

INSERT INTO sw_students2 (st_id, s_name)
VALUES (Id_seq.CURRVAL,'OMAR');

select * from sw_students2
```



ST_ID	S_NAME	S_DOB
1	ali	(null)
2	OSMAN	(null)
3	OMAR	(null)

```
INSERT INTO sw_students2 (st_id, s_name)
VALUES (Id_seq.CURRVAL, 'OSMAN');
```

```
INSERT INTO sw_students2 (st_id, s_name)
VALUES (Id_seq.CURRVAL, 'OMAR');
```

```
select * from sw_students2
```

Results Script Output Explain Autotrace DBMS

results:

	ST_ID	S_NAME	S_DOB
1	40	ali	(null)
2	40	OSMAN	(null)
3	40	OMAR	(null)

```
INSERT INTO sw_students2 (st_id, s_name)
VALUES (Id_seq.NEXTVAL, 'OSMAN');
```

```
INSERT INTO sw_students2 (st_id, s_name)
VALUES (Id_seq.NEXTVAL, 'OMAR');
```

```
select * from sw_students2
```

Results Script Output Explain Autotrace DBMS

results:

	ST_ID	S_NAME	S_DOB
1	40	ali	(null)
2	40	OSMAN	(null)
3	40	OMAR	(null)
4	50	OSMAN	(null)
5	60	OMAR	(null)

USES AND RESTRICTIONS OF NEXTVAL AND CURRVAL

CURRVAL and NEXTVAL can be used in the following places:

- VALUES clause of INSERT statements.
- The SELECT list of a SELECT statement.
- The SET clause of an UPDATE statement.

CURRVAL and NEXTVAL cannot be used in these places:

- A subquery.
- A view query or materialized view query.
- A SELECT statement with the DISTINCT operator.
- A SELECT statement with a GROUP BY or ORDER BY clause.
- The WHERE clause of a SELECT statement.
- DEFAULT value of a column in a CREATE TABLE or ALTER TABLE statement.
- The condition of a CHECK constraint.

USING THE SEQUENCE VIA THE IDENTITY COLUMN

- From Oracle 12c, you can associate a sequence with a table column via the identity column.
- Behind the scenes, Oracle creates a sequence that associates with the identity column.
- Because Oracle generates the sequence automatically for the identity column, the name of the sequence is different.
- Oracle uses the sys.idnseq\$ to store the link between the table and the sequence.
- Every IDENTITY column you add to a table requires its own Sequence Generator (SG). The SG is responsible for several tasks, including obtaining and supplying values to the IDENTITY column as necessary.
- Each IDENTITY column requires an associated, dedicated Sequence Generator (SG). When you add an IDENTITY column, the system creates an SG that's runs on the client with the application.
- Information about all attributes for every SG is added to a system table, SYS\$SGAttributesTable.

EXAMPLE

```
CREATE TABLE data( id NUMBER GENERATED ALWAYS AS IDENTITY ,  
                    title VARCHAR2(25) NOT NULL);
```

IDENTITY Column Statement	Description
GENERATED ALWAYS AS IDENTITY	The sequence generator always supplies an IDENTITY value. You cannot specify a value for the column.
GENERATED BY DEFAULT AS IDENTITY	The sequence generator supplies an IDENTITY value any time you do not supply a column value.
GENERATED BY DEFAULT ON NULL AS IDENTITY	The sequence generator supplies the next IDENTITY value if you specify a NULL column value.

SEQUENCE IN DATA DICTIONARY

```
SELECT * FROM USER_SEQUENCES
```

	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
1	ID_SEQ	10	100	10	Y	N	2	70
2	ID_SEQ2	10	100	10	Y	N	2	30

```
SELECT * FROM ALL_SEQUENCES
```

[illegible]

ALTERING SEQUENCES

- To alter a sequence, your schema must contain the sequence, or you must have the ALTER ANY SEQUENCE system privilege.
- You can alter a sequence to change any of the parameters that define how it generates sequence numbers except the sequence starting number.
- To change the starting point of a sequence, drop the sequence and then re-create it.

SYNTAX:

```
ALTER SEQUENCE Id_seq  
INCREMENT BY 20  
MAXVALUE 1000  
NOCYCLE  
CACHE 20 ;
```

DROPPING SEQUENCE

- You can drop any sequence in your schema.
- To drop a sequence in another schema, you must have the DROP ANY SEQUENCE system privilege.
- If a sequence is no longer required, you can drop the sequence using the DROP SEQUENCE statement.
- When a sequence is dropped, its definition is removed from the data dictionary.
- Any synonyms for the sequence remain but return an error when referenced.

EXAMPLE

```
DROP SEQUENCE id_seq ;
```