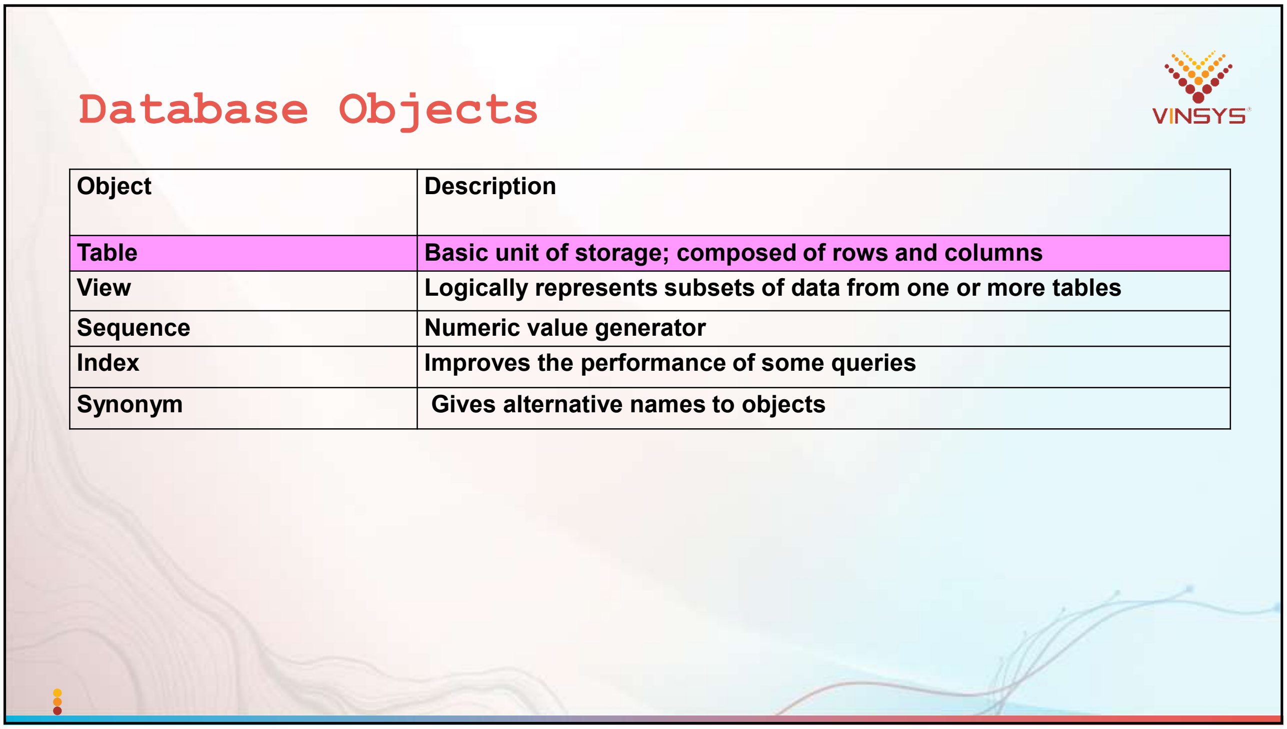
VINSYS

DEliVERiNGSKiLLS
DRiViNGSUCCESS

www.VINSYS.COM

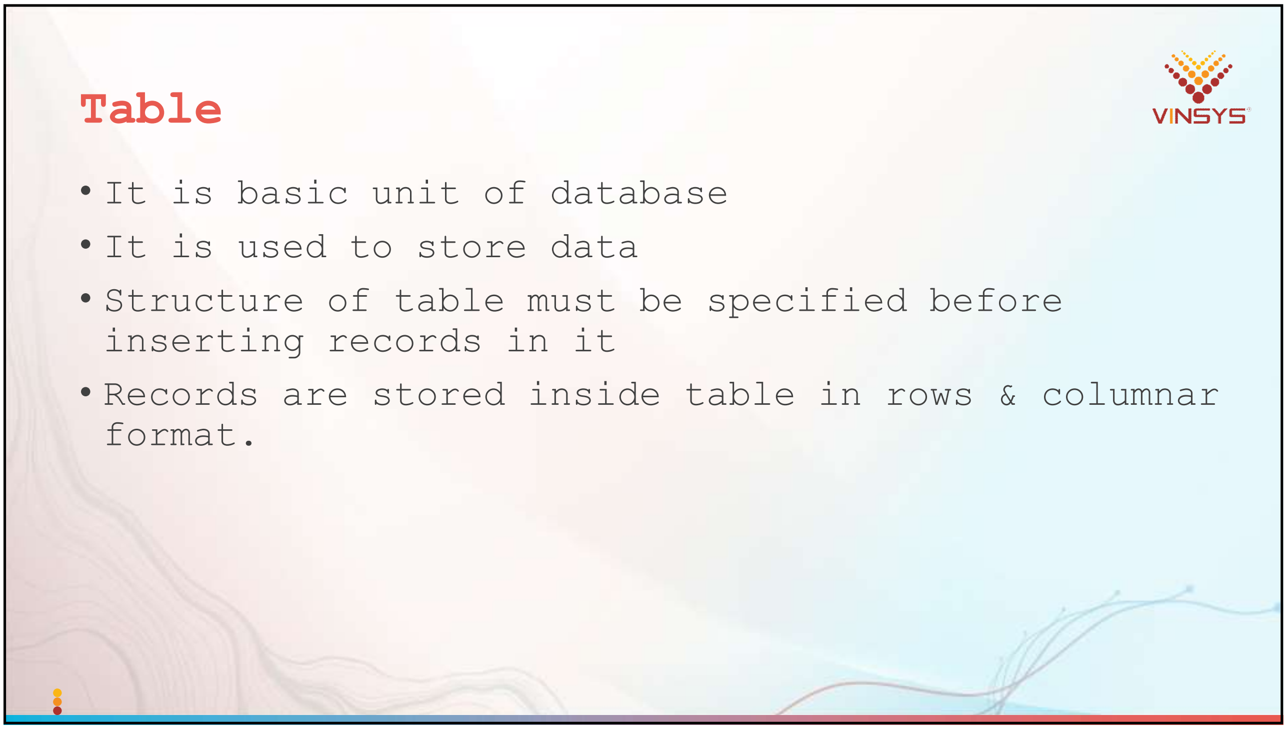
VINSYS

- Database Objects



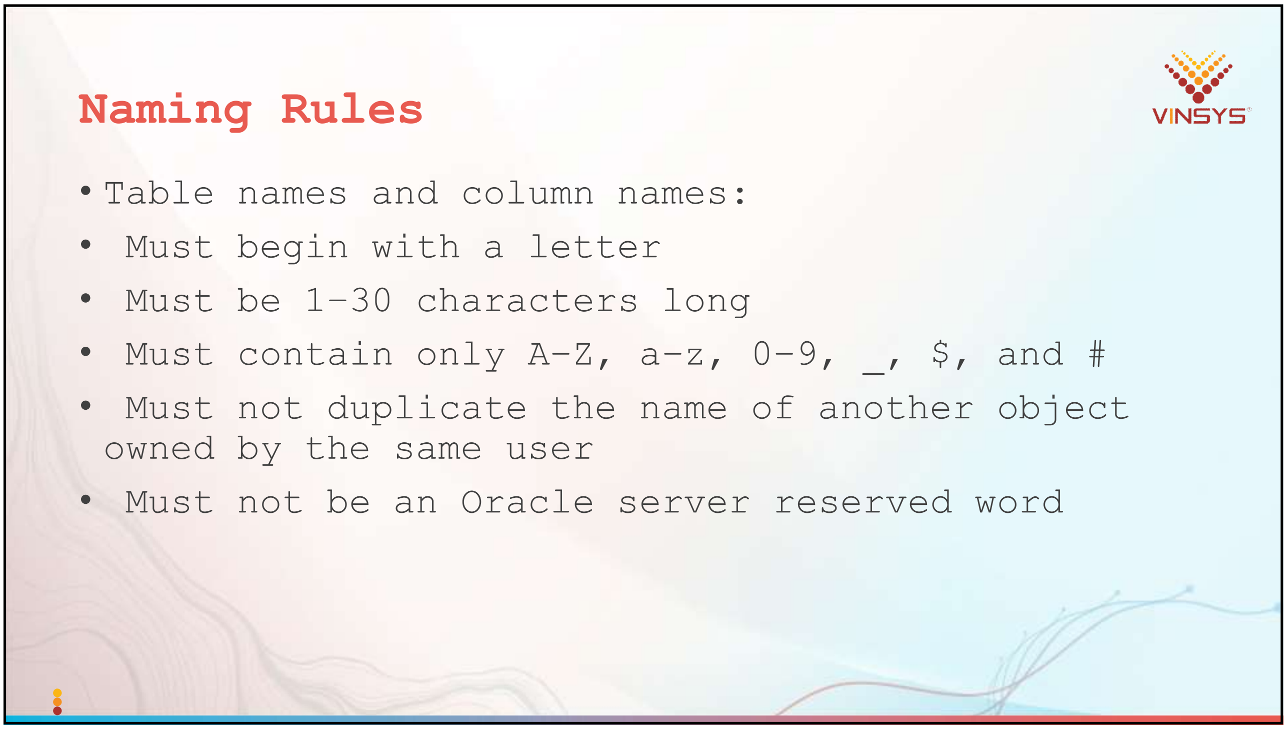
Database Objects

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Numeric value generator
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects



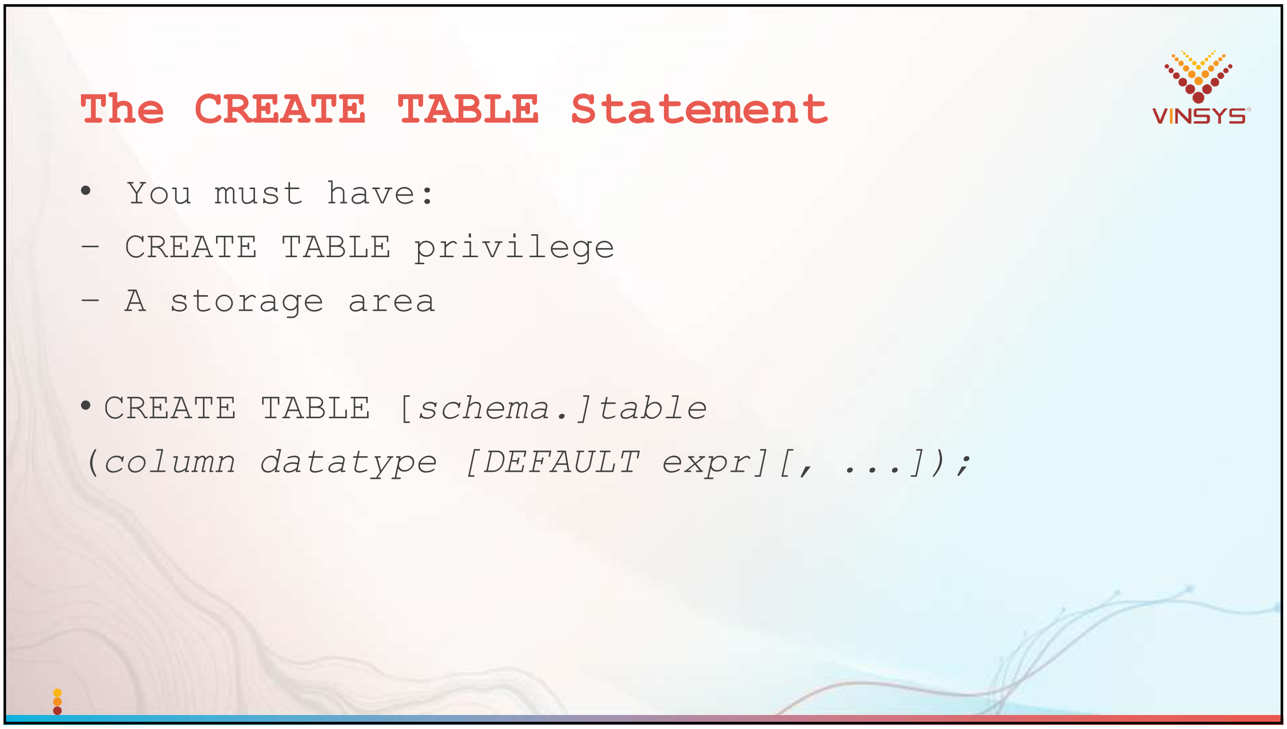
Table

- It is basic unit of database
- It is used to store data
- Structure of table must be specified before inserting records in it
- Records are stored inside table in rows & columnar format.



Naming Rules

- Table names and column names:
- Must begin with a letter
- Must be 1–30 characters long
- Must contain only A–Z, a–z, 0–9, _, $, and #
- Must not duplicate the name of another object owned by the same user
- Must not be an Oracle server reserved word



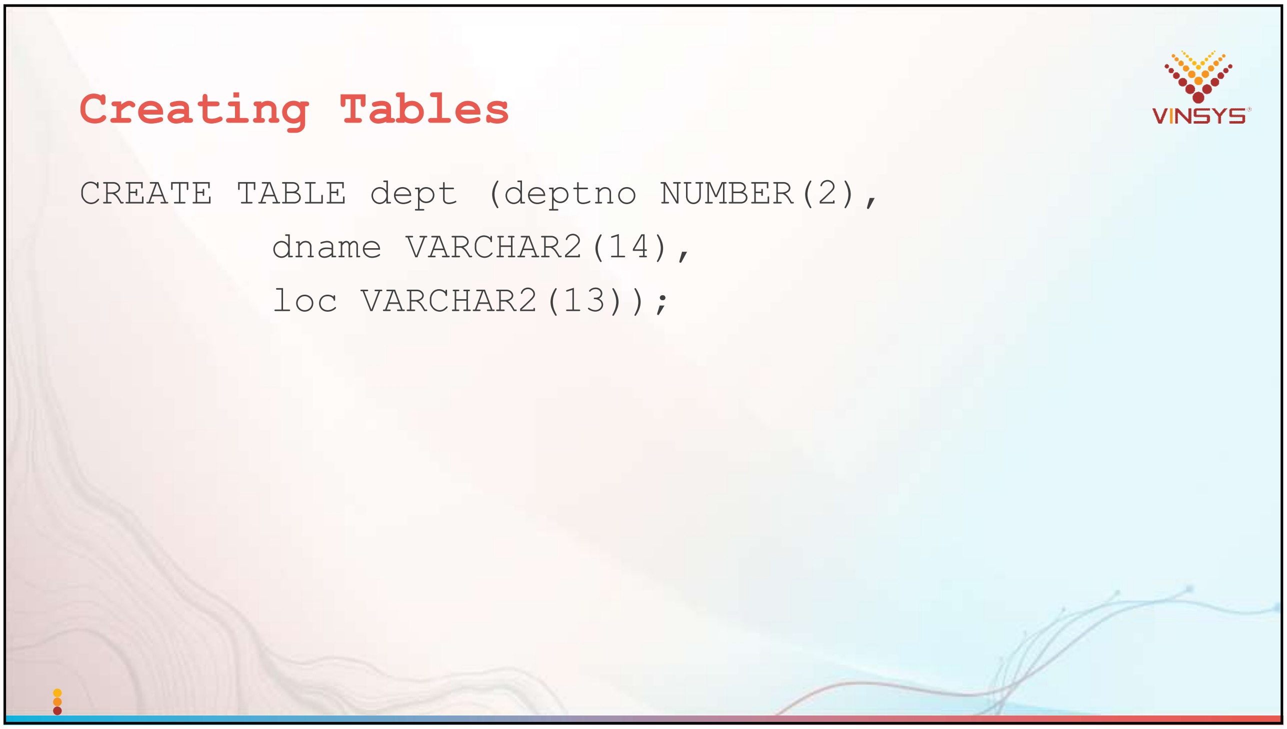
The CREATE TABLE Statement

- You must have:

– CREATE TABLE privilege

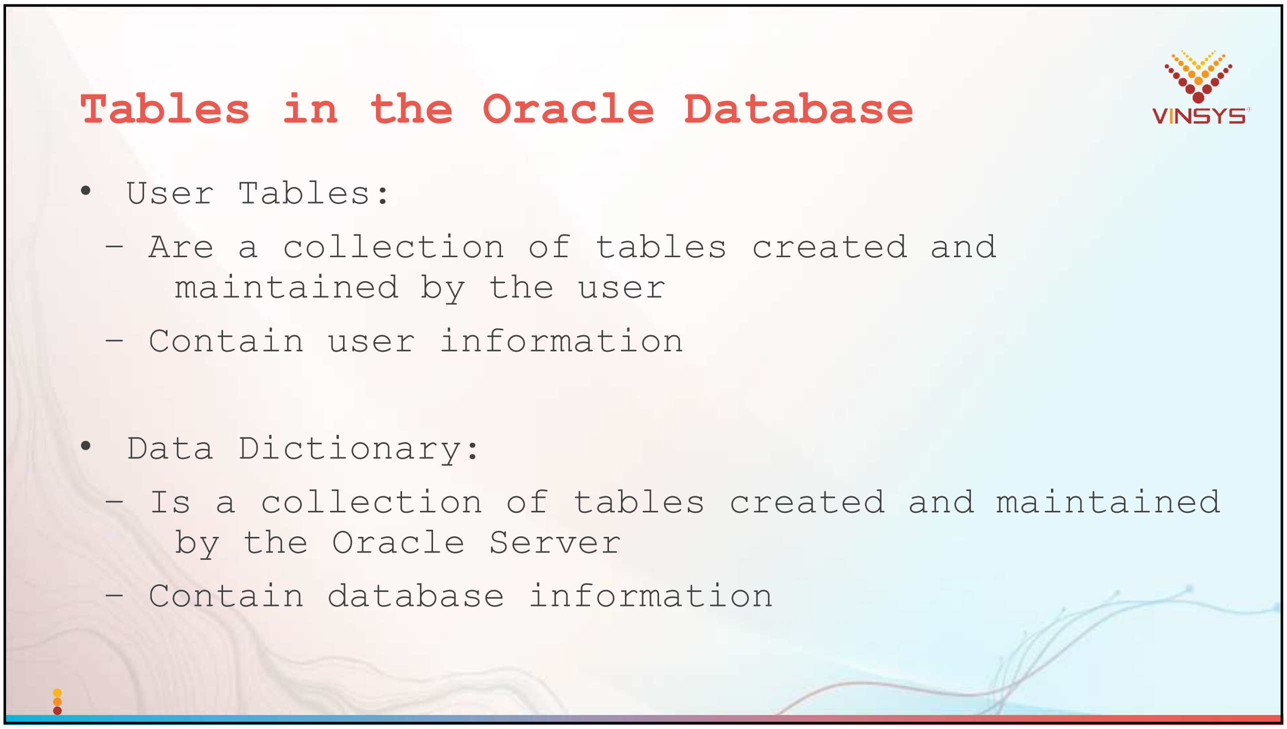
– A storage area

- CREATE TABLE [*schema.*]*table*
(*column datatype* [DEFAULT *expr*][, ...]);



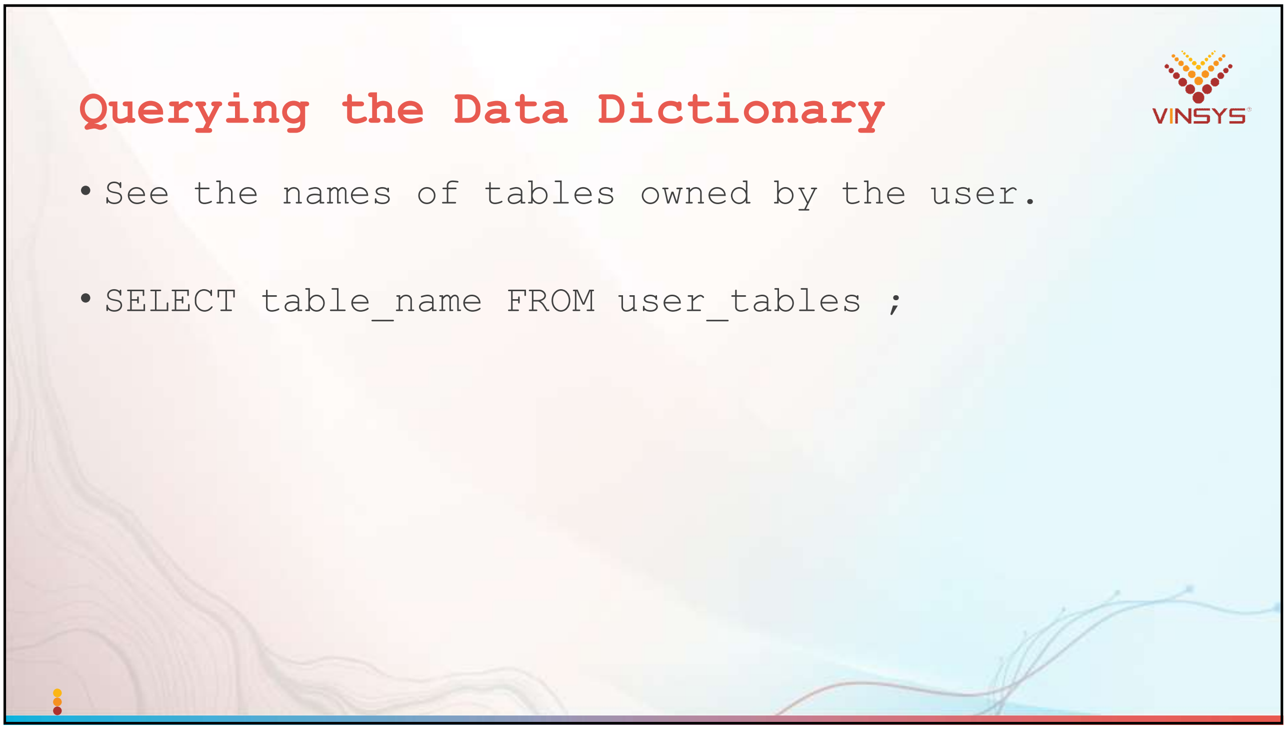
Creating Tables

```
CREATE TABLE dept (deptno NUMBER(2),
          dname VARCHAR2(14),
          loc VARCHAR2(13));
```



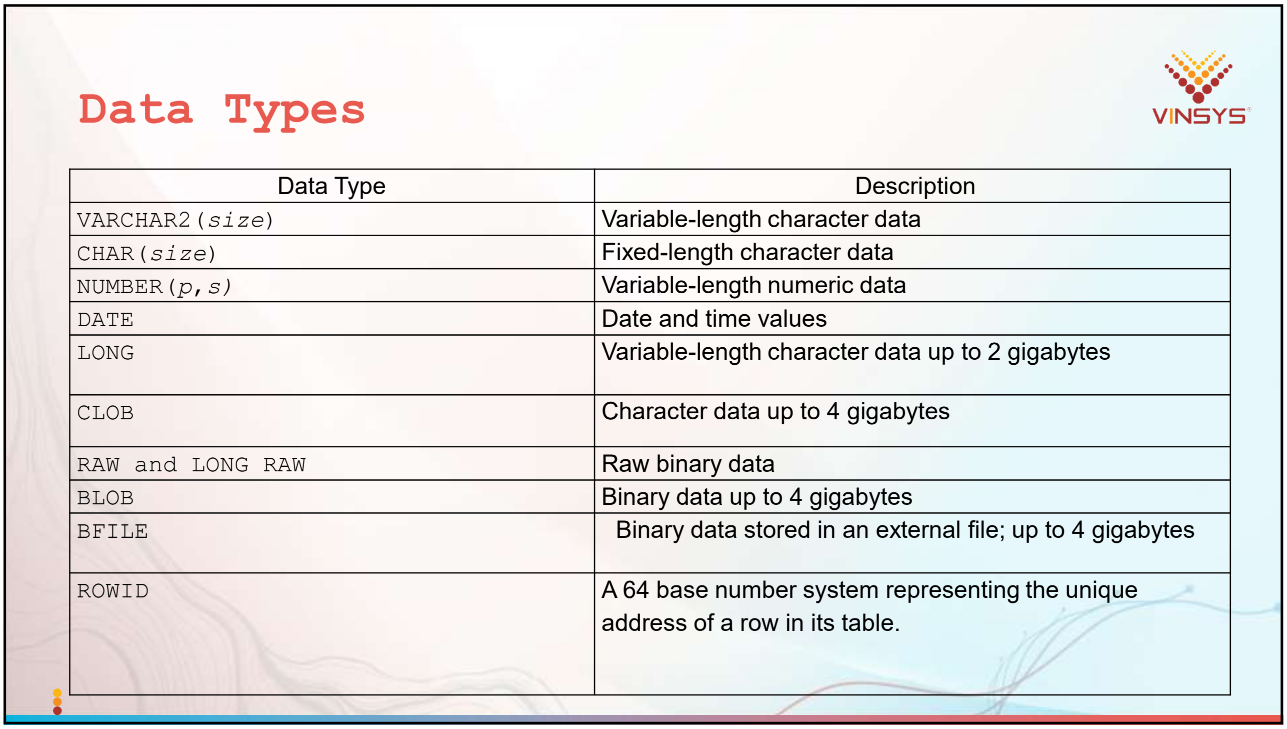
Tables in the Oracle Database

- User Tables:
  - Are a collection of tables created and maintained by the user
  - Contain user information
- Data Dictionary:
  - Is a collection of tables created and maintained by the Oracle Server
  - Contain database information



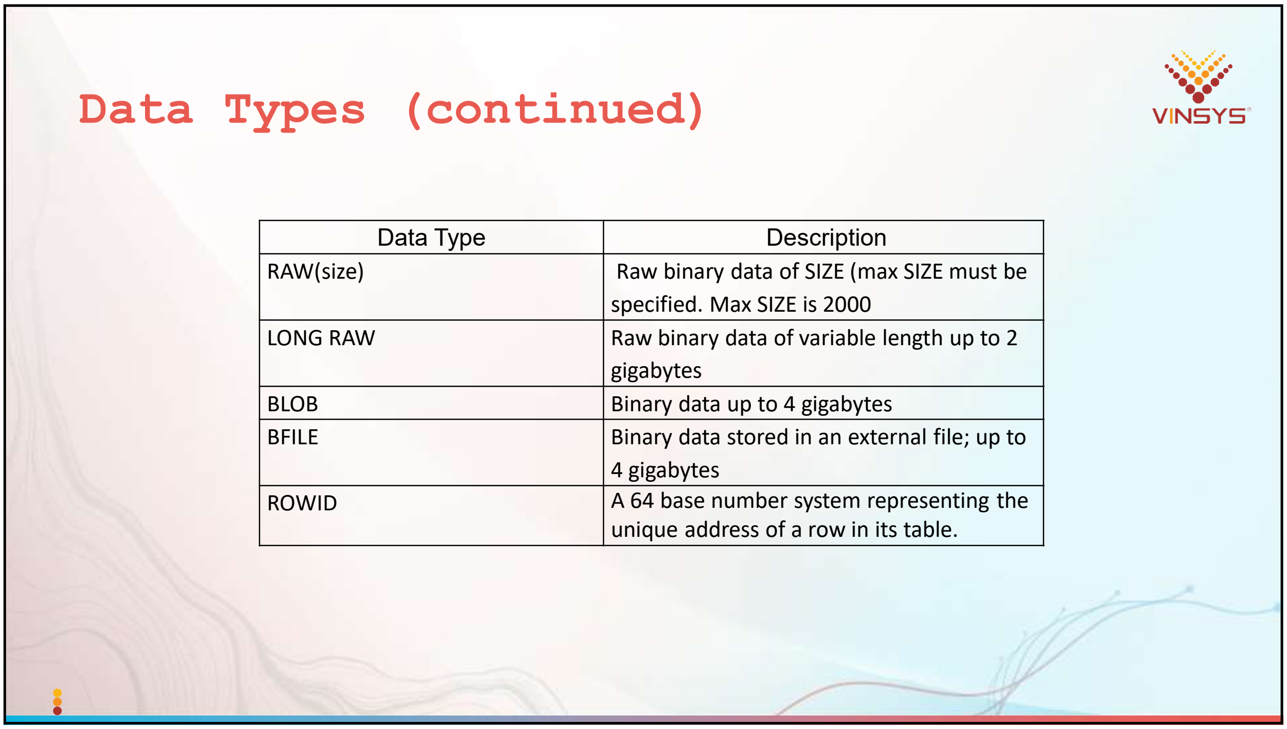
Querying the Data Dictionary

- See the names of tables owned by the user.
- SELECT table_name FROM user_tables ;



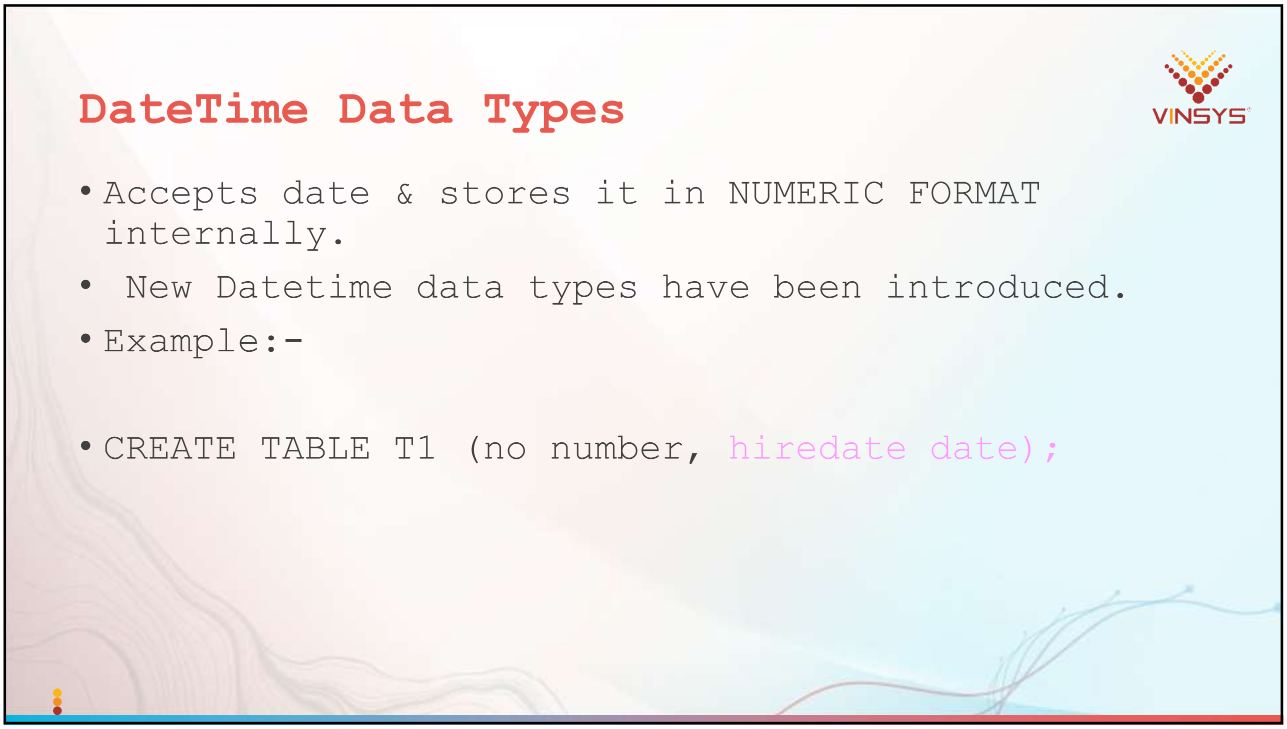
Data Types

Data Type	Description
VARCHAR2(*size*)	Variable-length character data
CHAR(*size*)	Fixed-length character data
NUMBER(*p,s*)	Variable-length numeric data
DATE	Date and time values
LONG	Variable-length character data up to 2 gigabytes
CLOB	Character data up to 4 gigabytes
RAW and LONG RAW	Raw binary data
BLOB	Binary data up to 4 gigabytes
BFILE	Binary data stored in an external file; up to 4 gigabytes
ROWID	A 64 base number system representing the unique address of a row in its table.



Data Types (continued)

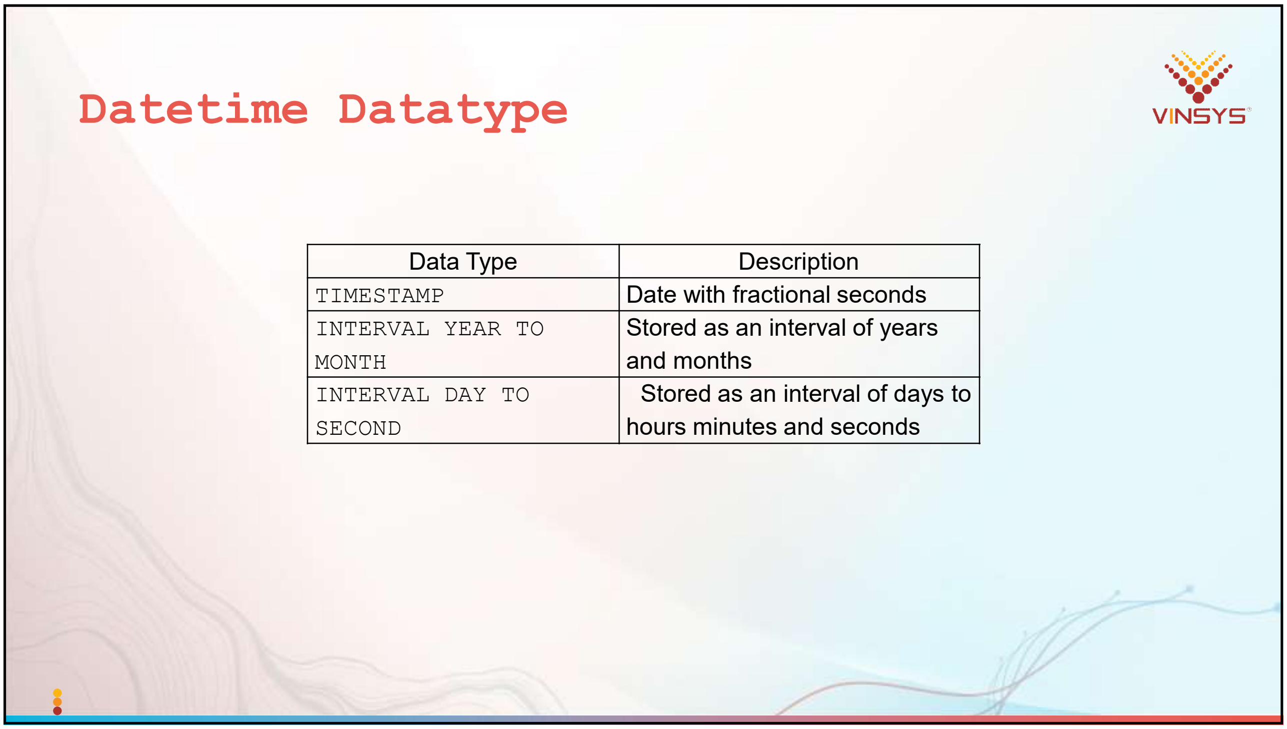
Data Type	Description
RAW(size)	Raw binary data of SIZE (max SIZE must be specified. Max SIZE is 2000
LONG RAW	Raw binary data of variable length up to 2 gigabytes
BLOB	Binary data up to 4 gigabytes
BFILE	Binary data stored in an external file; up to 4 gigabytes
ROWID	A 64 base number system representing the unique address of a row in its table.



DateTime Data Types

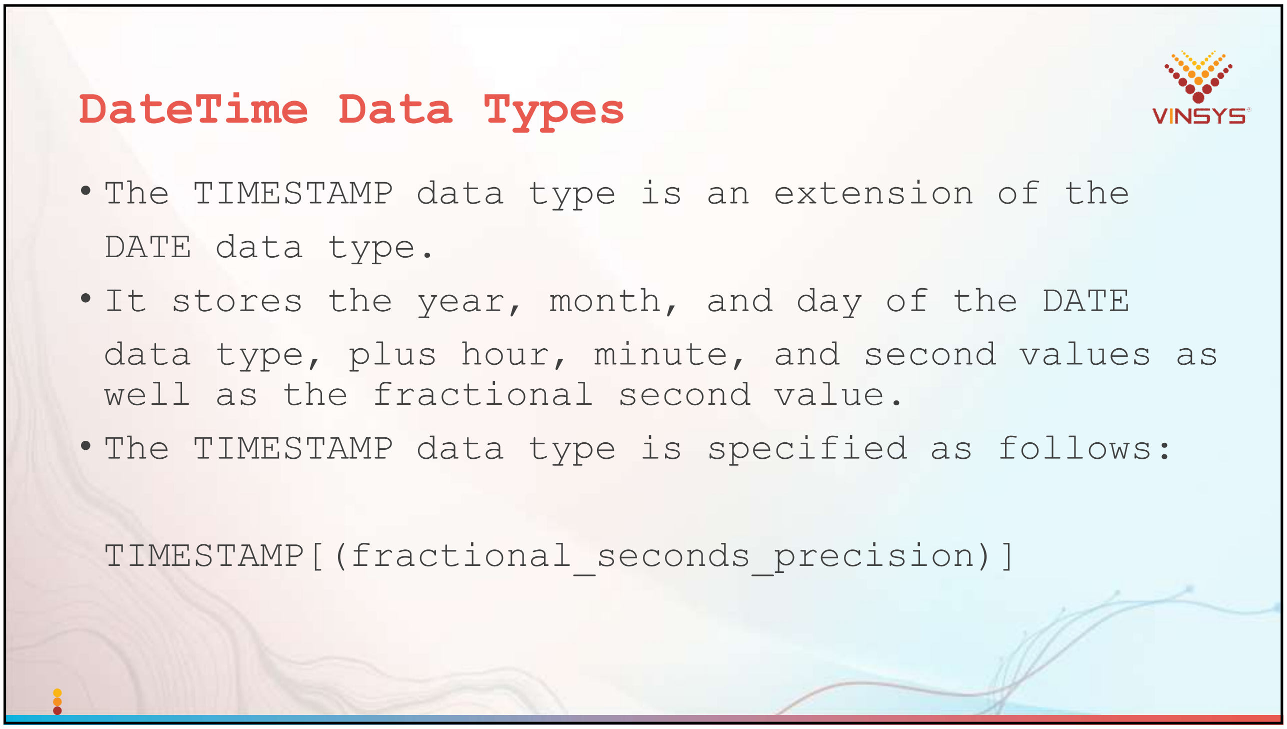
- Accepts date & stores it in NUMERIC FORMAT internally.
- New Datetime data types have been introduced.
- Example:-

- CREATE TABLE T1 (no number, hiredate date);



Datetime Datatype

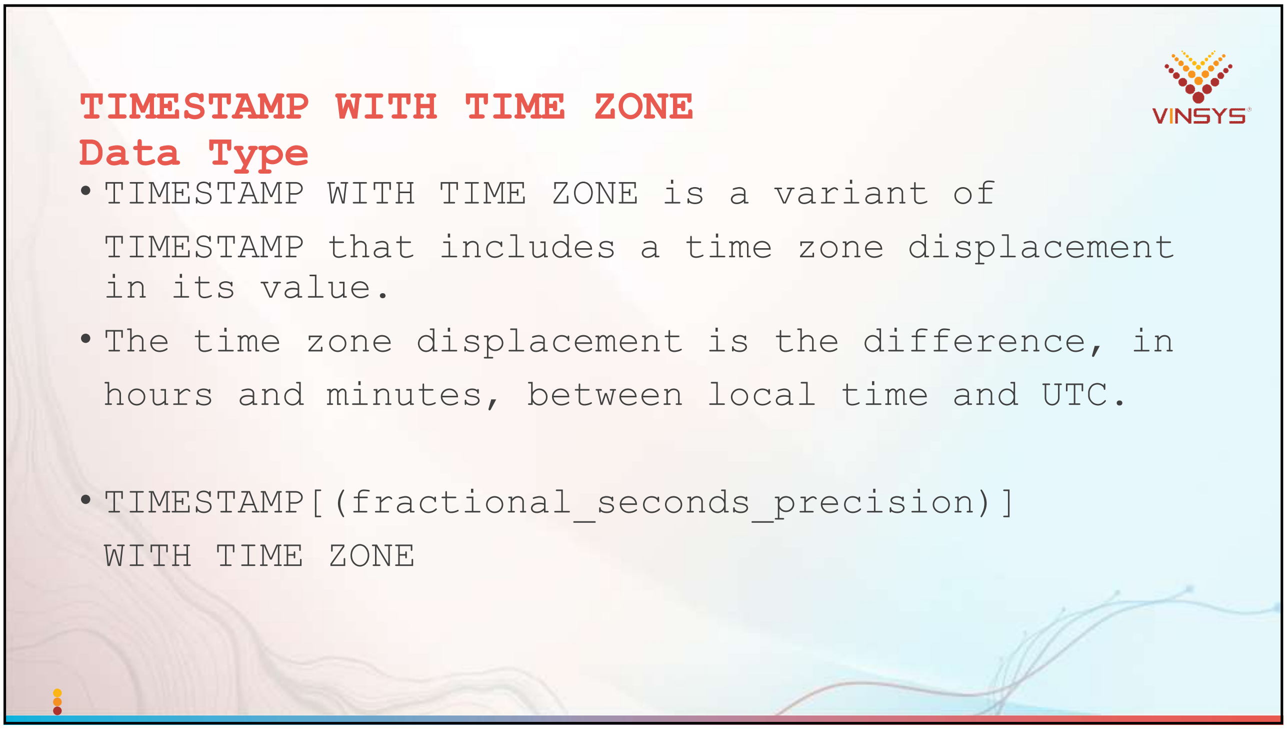
Data Type	Description
TIMESTAMP	Date with fractional seconds
INTERVAL YEAR TO MONTH	Stored as an interval of years and months
INTERVAL DAY TO SECOND	Stored as an interval of days to hours minutes and seconds



DateTime Data Types

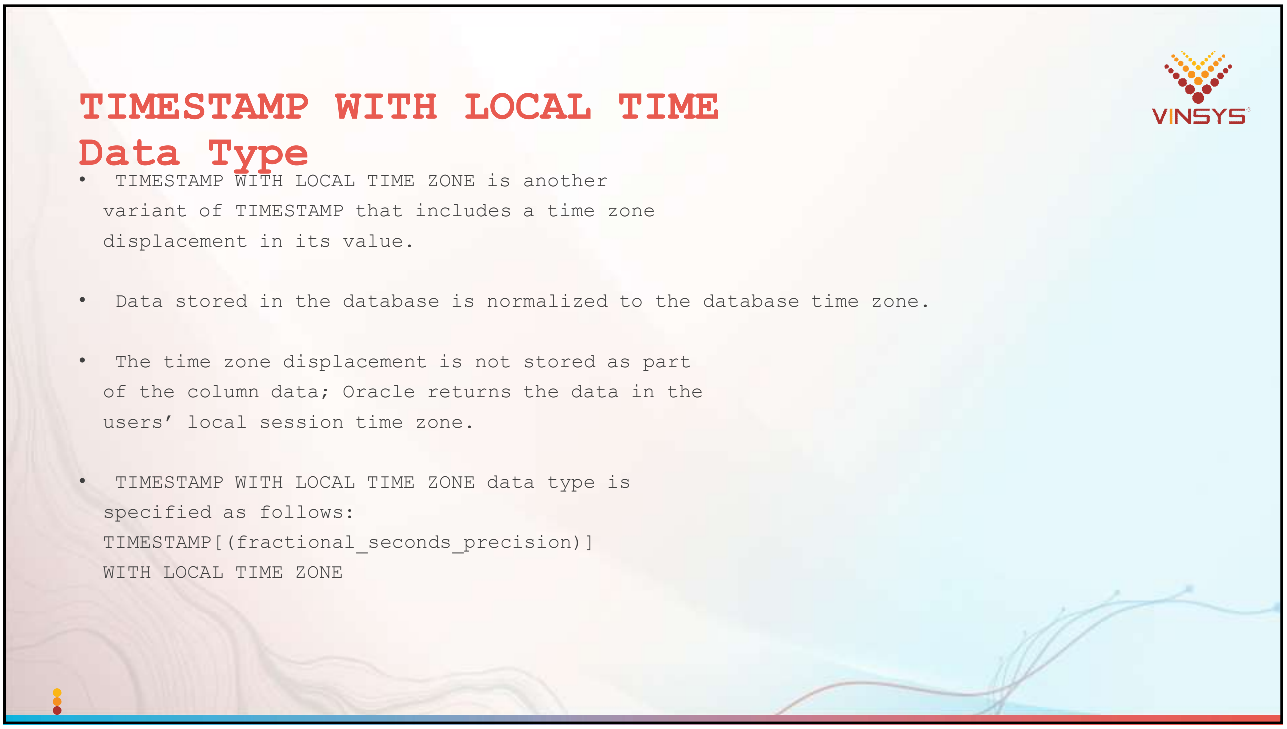
- The TIMESTAMP data type is an extension of the DATE data type.
- It stores the year, month, and day of the DATE data type, plus hour, minute, and second values as well as the fractional second value.
- The TIMESTAMP data type is specified as follows:

```
TIMESTAMP[(fractional_seconds_precision)]
```



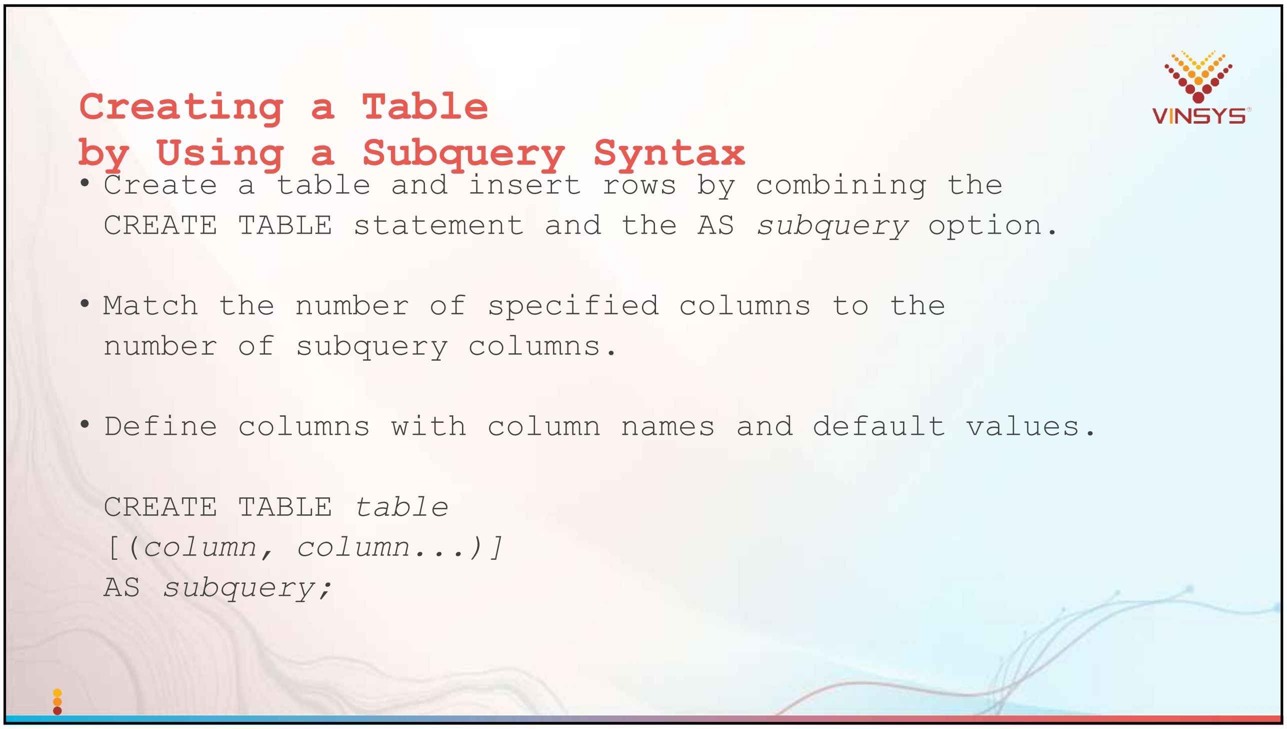
TIMESTAMP WITH TIME ZONE Data Type

- TIMESTAMP WITH TIME ZONE is a variant of TIMESTAMP that includes a time zone displacement in its value.
- The time zone displacement is the difference, in hours and minutes, between local time and UTC.
- TIMESTAMP[(fractional_seconds_precision)] WITH TIME ZONE



TIMESTAMP WITH LOCAL TIME Data Type

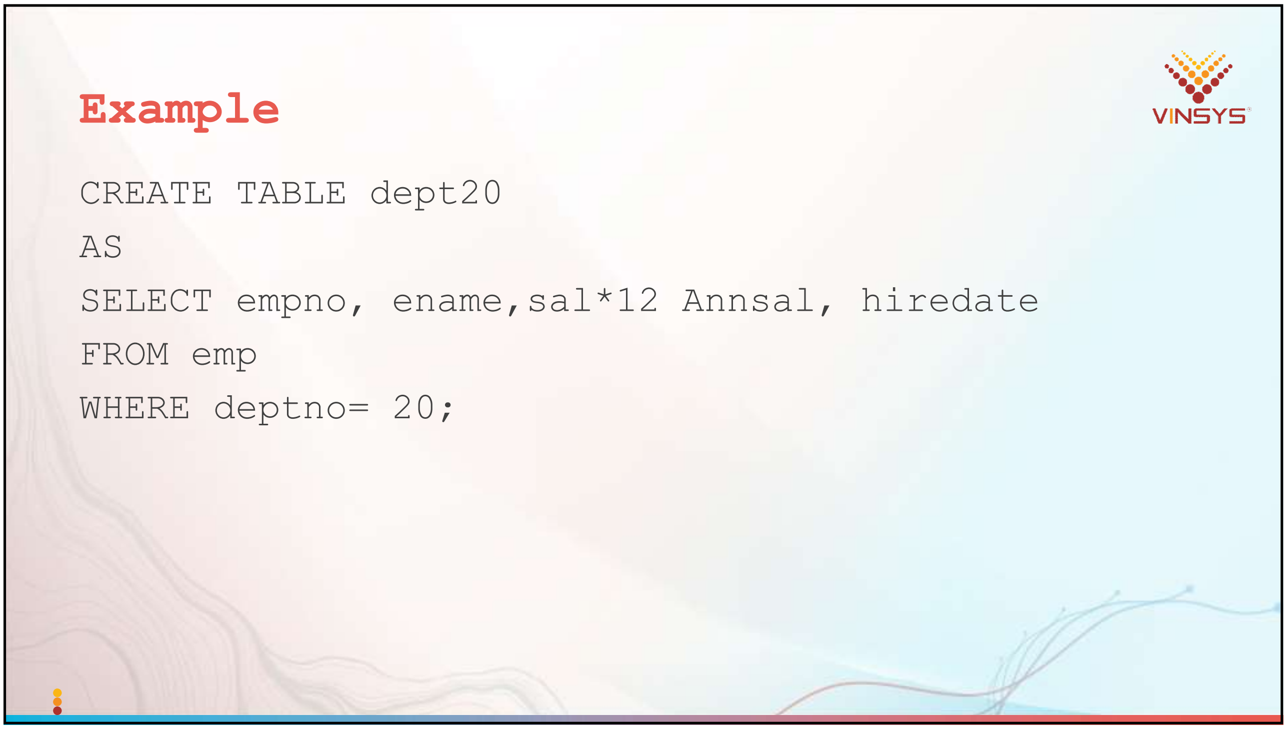
- TIMESTAMP WITH LOCAL TIME ZONE is another variant of TIMESTAMP that includes a time zone displacement in its value.
- Data stored in the database is normalized to the database time zone.
- The time zone displacement is not stored as part of the column data; Oracle returns the data in the users' local session time zone.
- TIMESTAMP WITH LOCAL TIME ZONE data type is specified as follows:
  TIMESTAMP[(fractional_seconds_precision)]
  WITH LOCAL TIME ZONE



Creating a Table by Using a Subquery Syntax

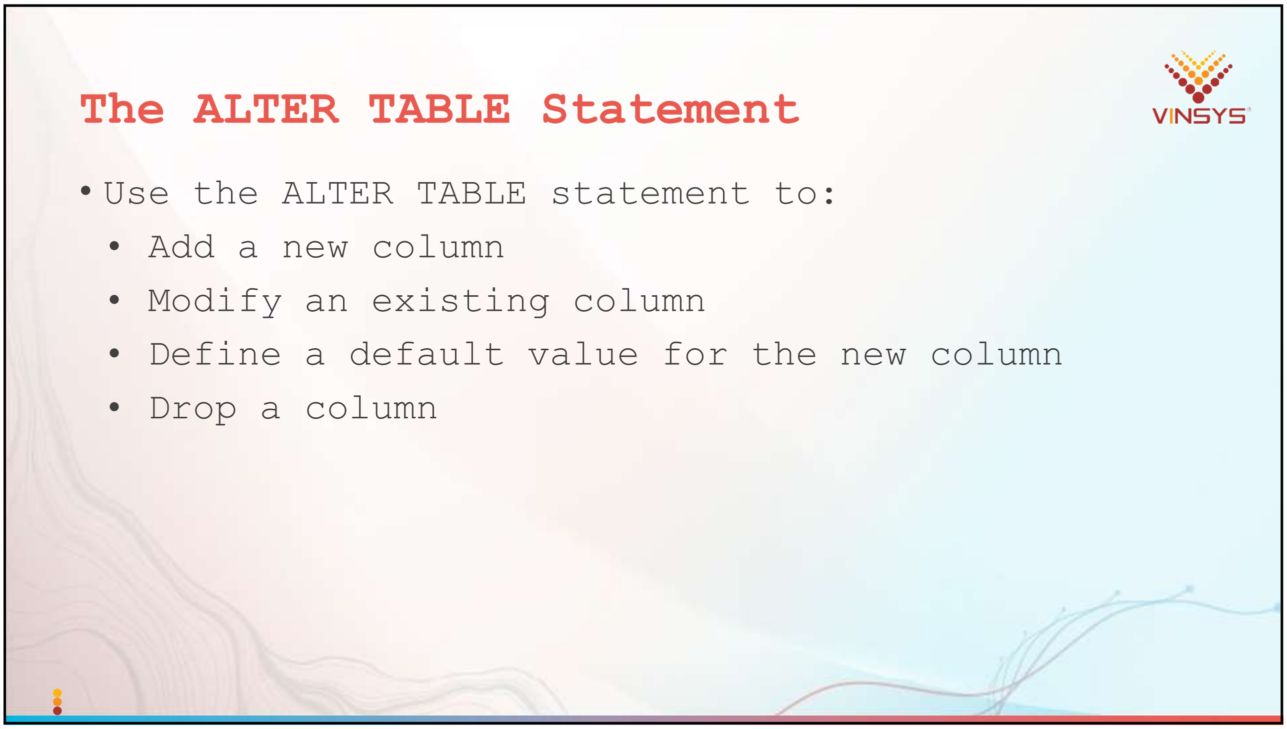
- Create a table and insert rows by combining the CREATE TABLE statement and the AS *subquery* option.
- Match the number of specified columns to the number of subquery columns.
- Define columns with column names and default values.

```
CREATE TABLE table
[(column, column...)]
AS subquery;
```



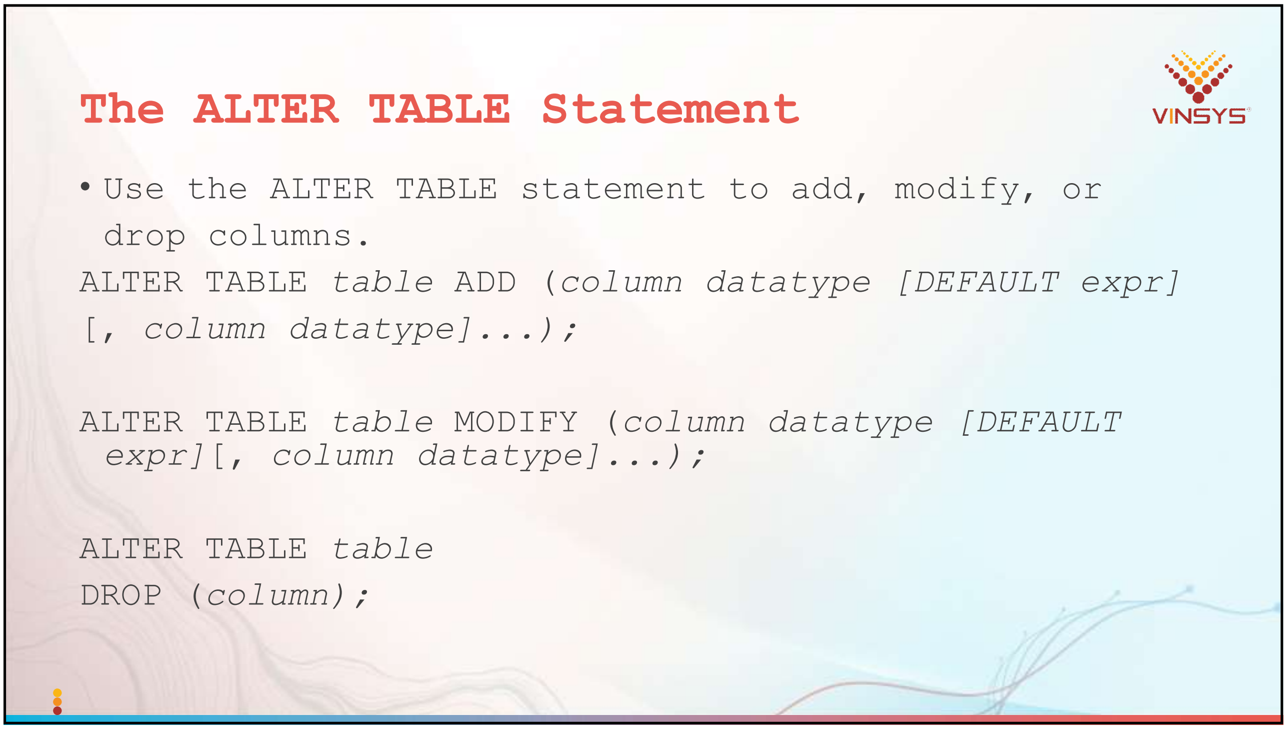
Example

```
CREATE TABLE dept20
AS
SELECT empno, ename,sal*12 Annsal, hiredate
FROM emp
WHERE deptno= 20;
```



The ALTER TABLE Statement

- Use the ALTER TABLE statement to:
  - Add a new column
  - Modify an existing column
  - Define a default value for the new column
  - Drop a column



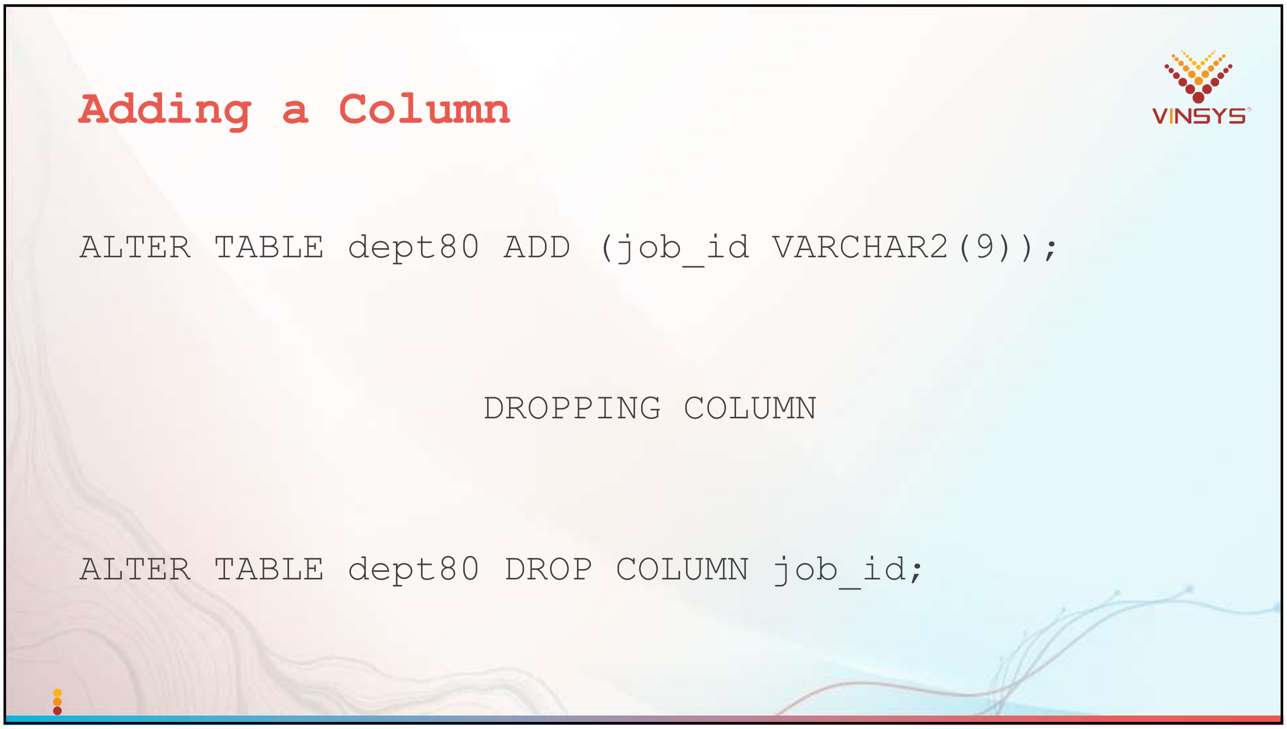
The ALTER TABLE Statement

- Use the ALTER TABLE statement to add, modify, or drop columns.

ALTER TABLE *table* ADD (*column datatype [DEFAULT expr]*
[, column datatype]...);

ALTER TABLE *table* MODIFY (*column datatype [DEFAULT expr][, column datatype]...);*

ALTER TABLE *table*
DROP (*column);*

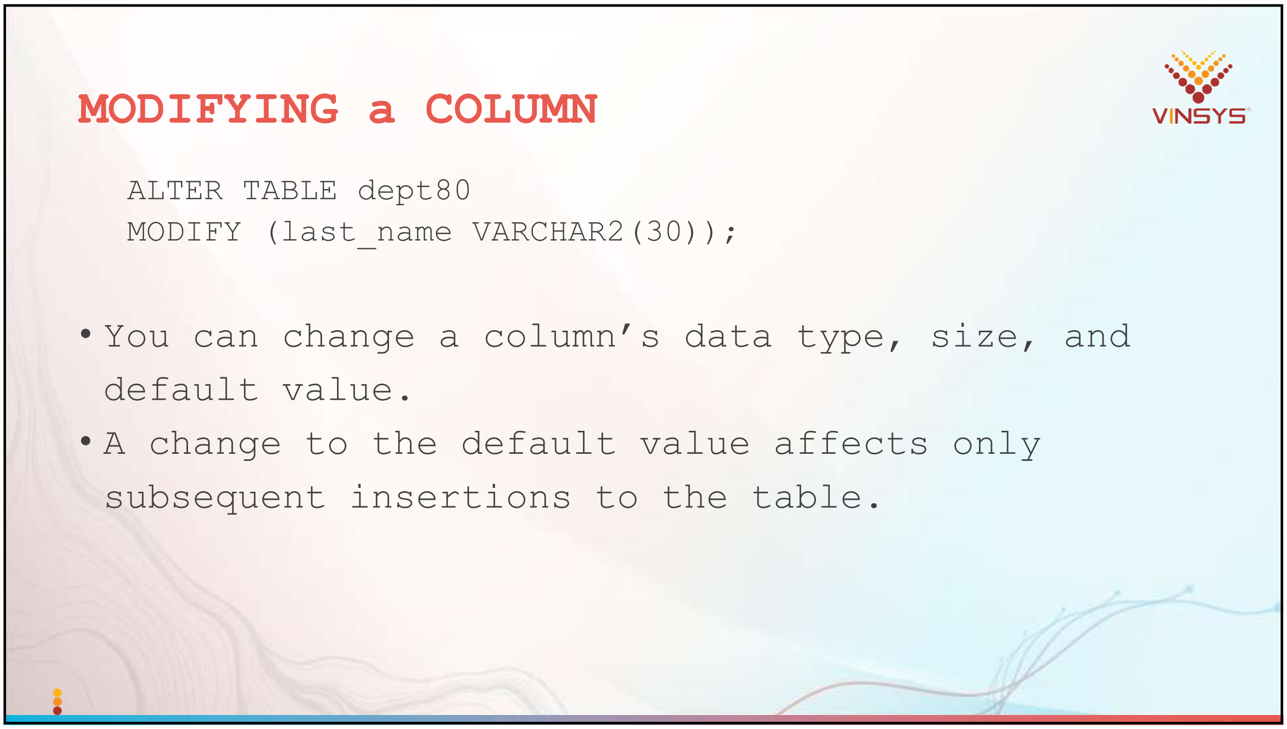


Adding a Column

```
ALTER TABLE dept80 ADD (job_id VARCHAR2(9));
```

DROPPING COLUMN

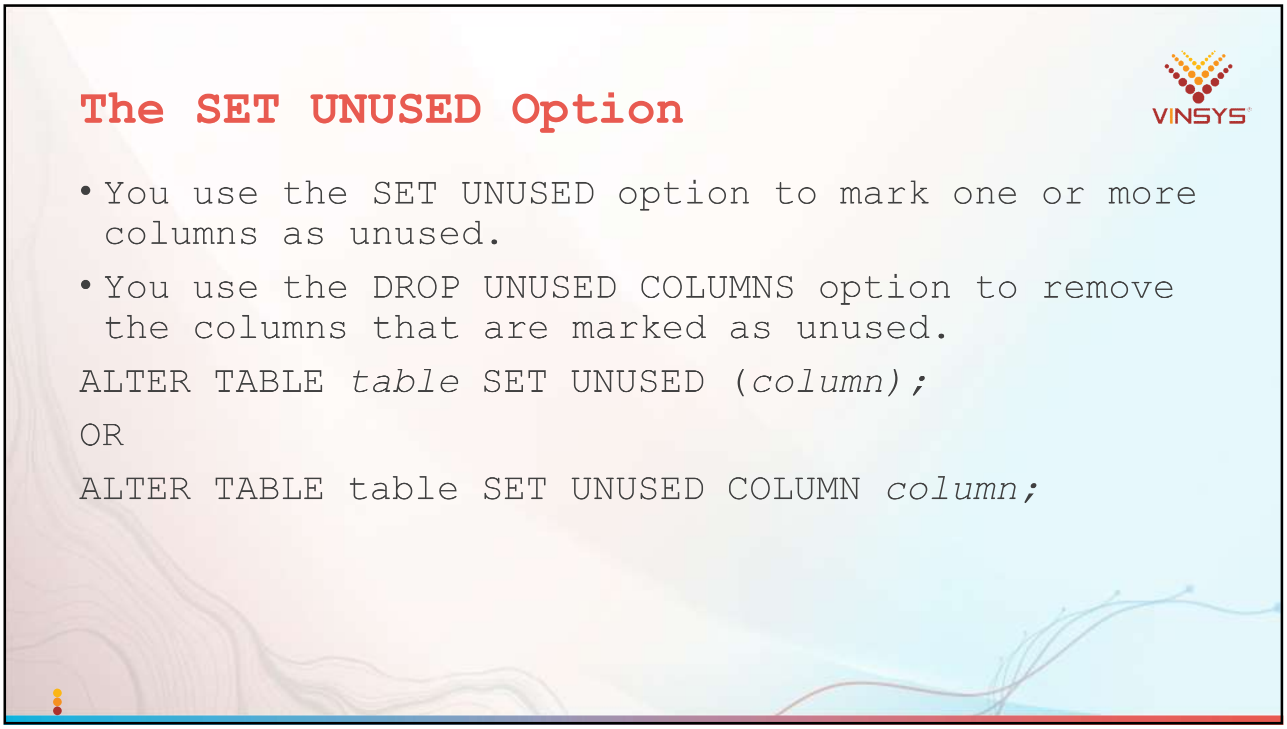
```
ALTER TABLE dept80 DROP COLUMN job_id;
```



MODIFYING a COLUMN

```
ALTER TABLE dept80
MODIFY (last_name VARCHAR2(30));
```

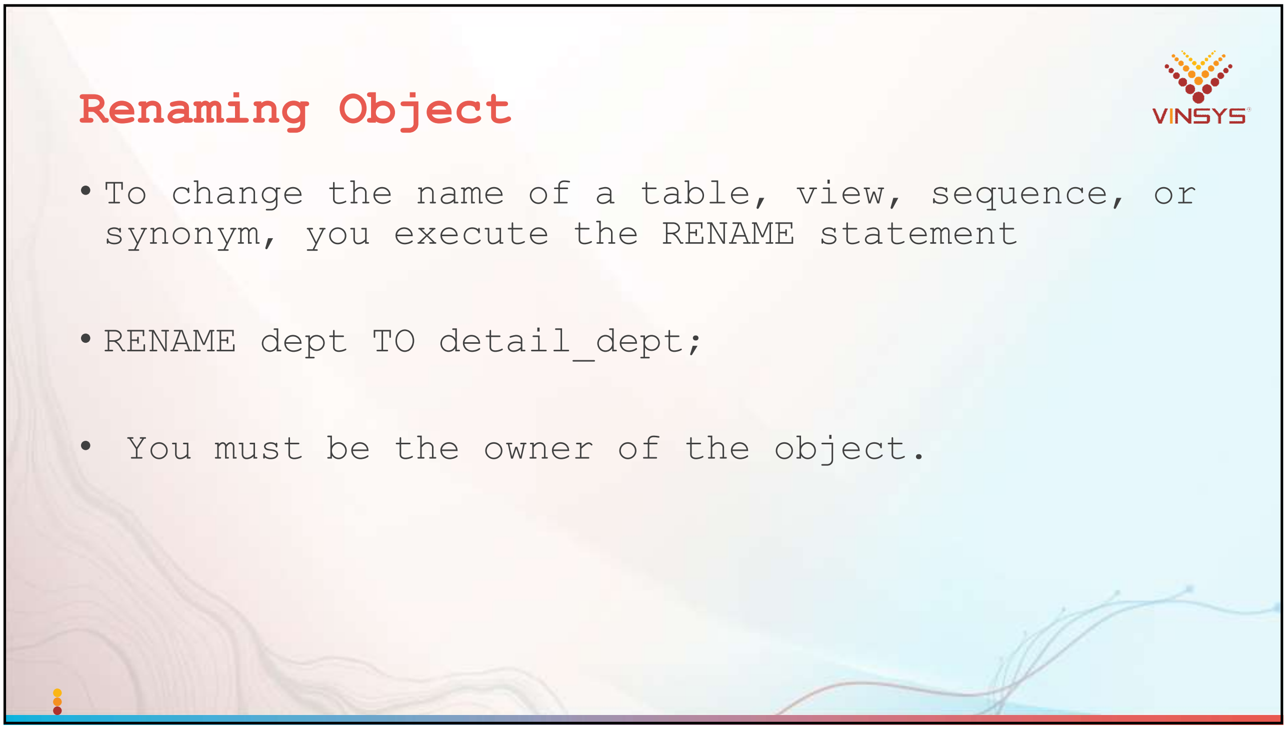
- You can change a column’s data type, size, and default value.
- A change to the default value affects only subsequent insertions to the table.



The SET UNUSED Option

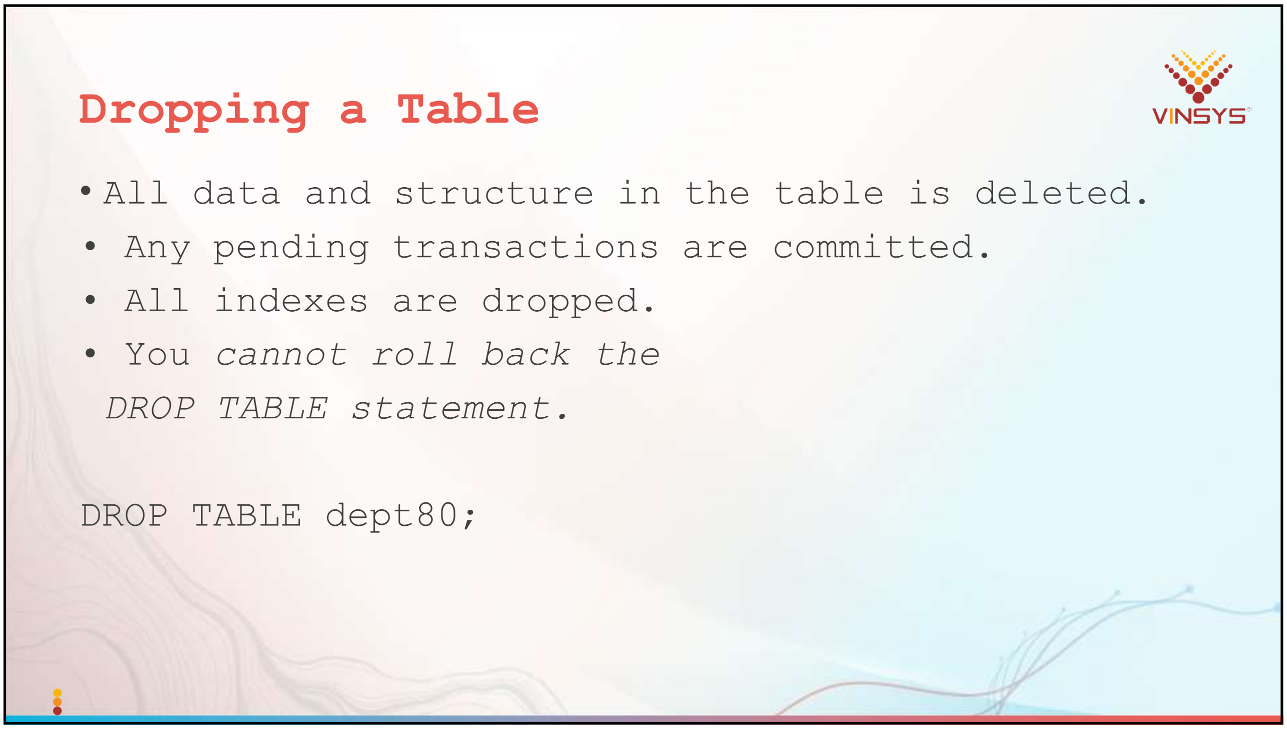
- You use the SET UNUSED option to mark one or more columns as unused.
- You use the DROP UNUSED COLUMNS option to remove the columns that are marked as unused.

```
ALTER TABLE table SET UNUSED (column);
OR
ALTER TABLE table SET UNUSED COLUMN column;
```



Renaming Object

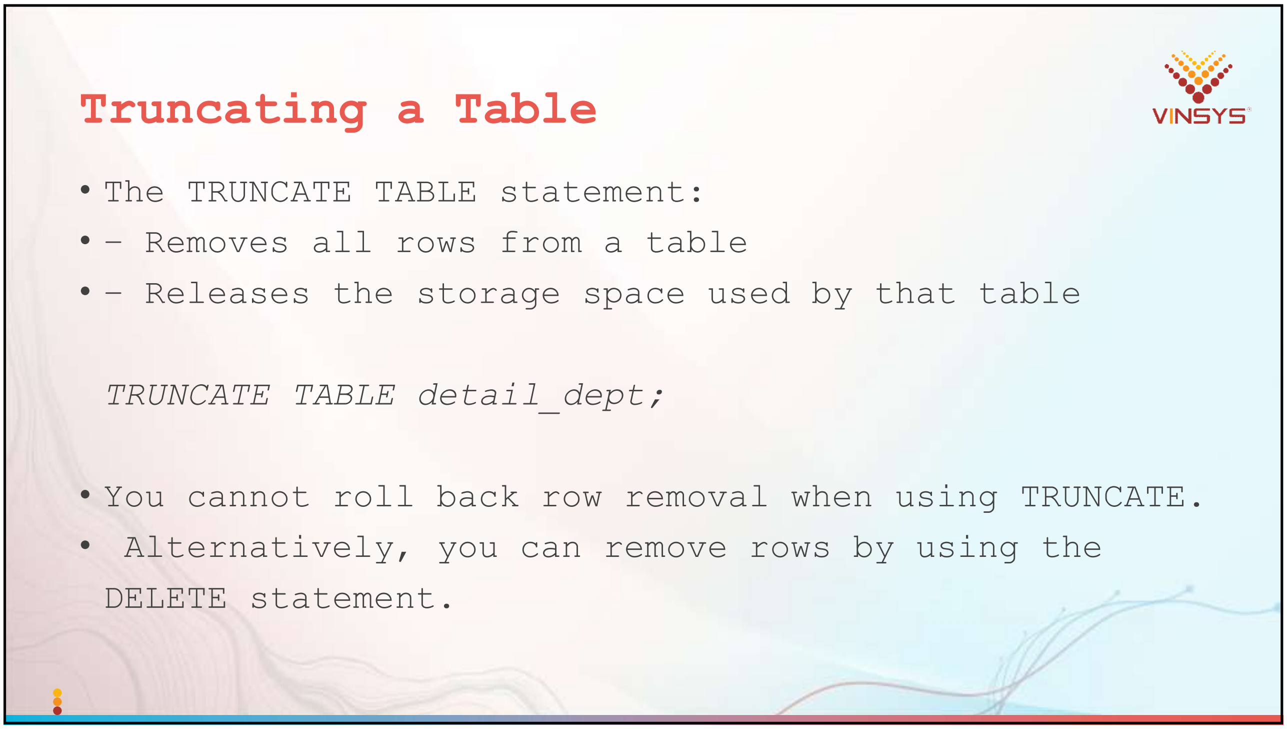
- To change the name of a table, view, sequence, or synonym, you execute the RENAME statement
- RENAME dept TO detail_dept;
- You must be the owner of the object.



Dropping a Table

- All data and structure in the table is deleted.
- Any pending transactions are committed.
- All indexes are dropped.
- You *cannot roll back the DROP TABLE statement.*

```
DROP TABLE dept80;
```

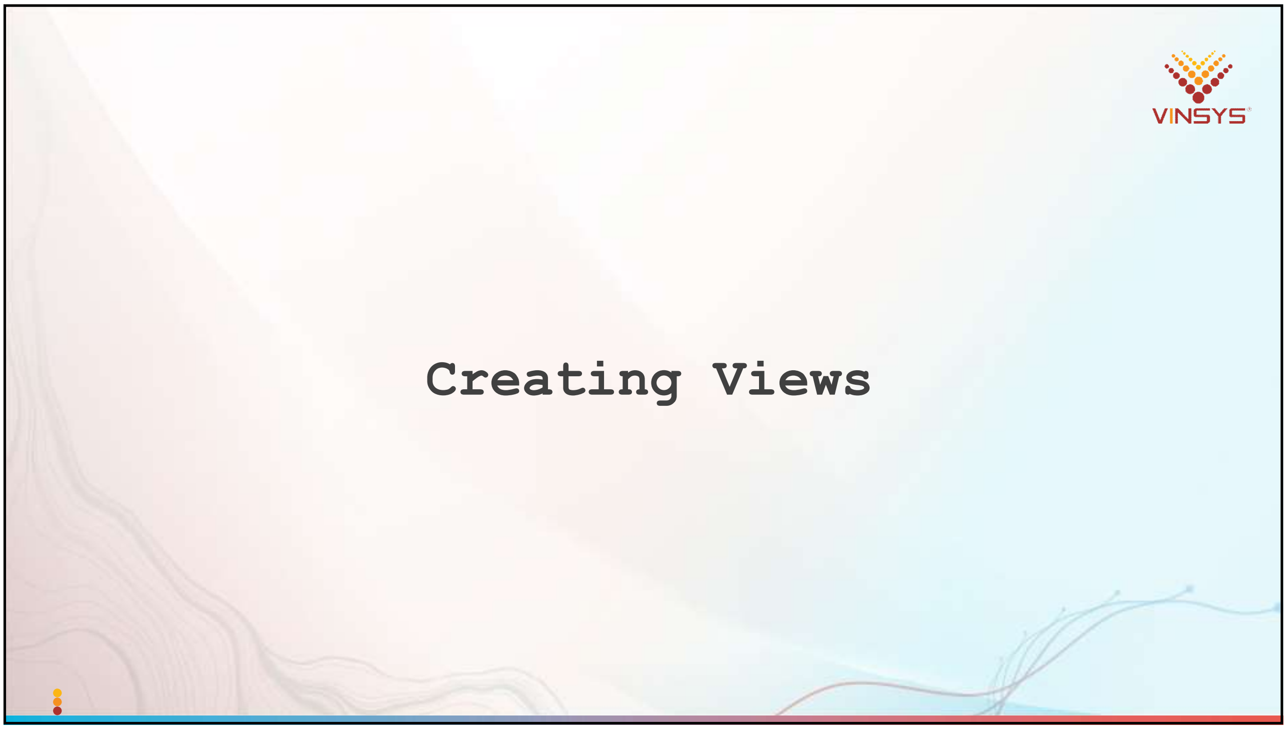


Truncating a Table

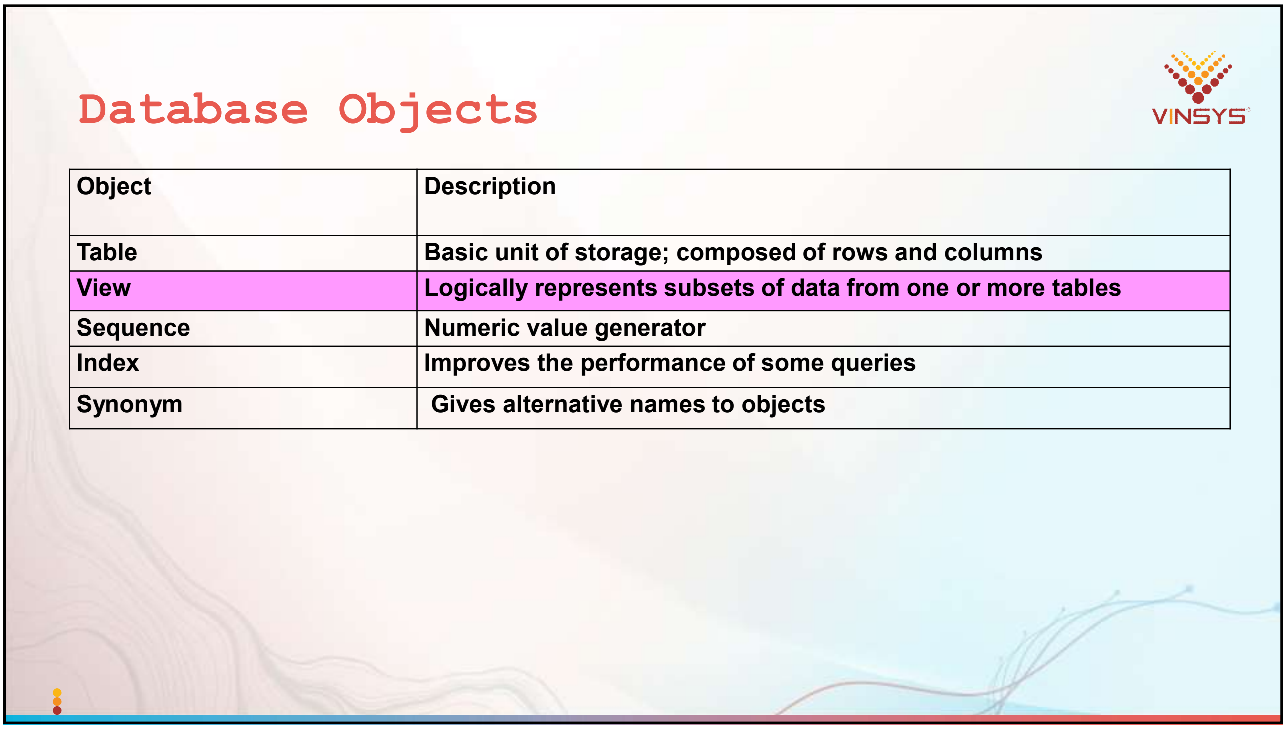
- The TRUNCATE TABLE statement:
- – Removes all rows from a table
- – Releases the storage space used by that table

```
TRUNCATE TABLE detail_dept;
```

- You cannot roll back row removal when using TRUNCATE.
- Alternatively, you can remove rows by using the DELETE statement.



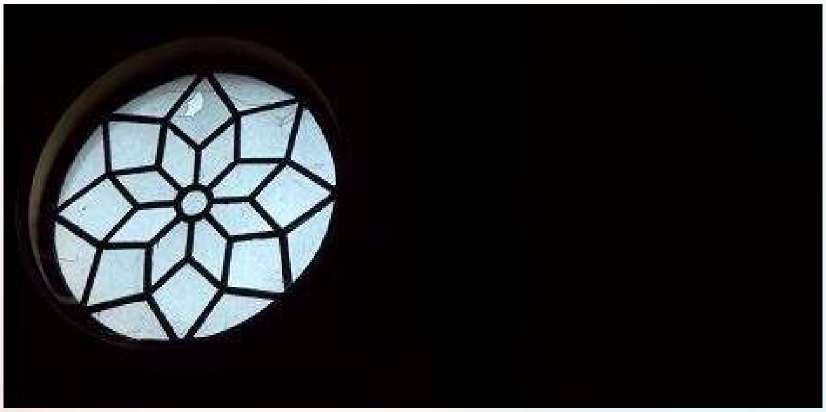
Creating Views



Database Objects

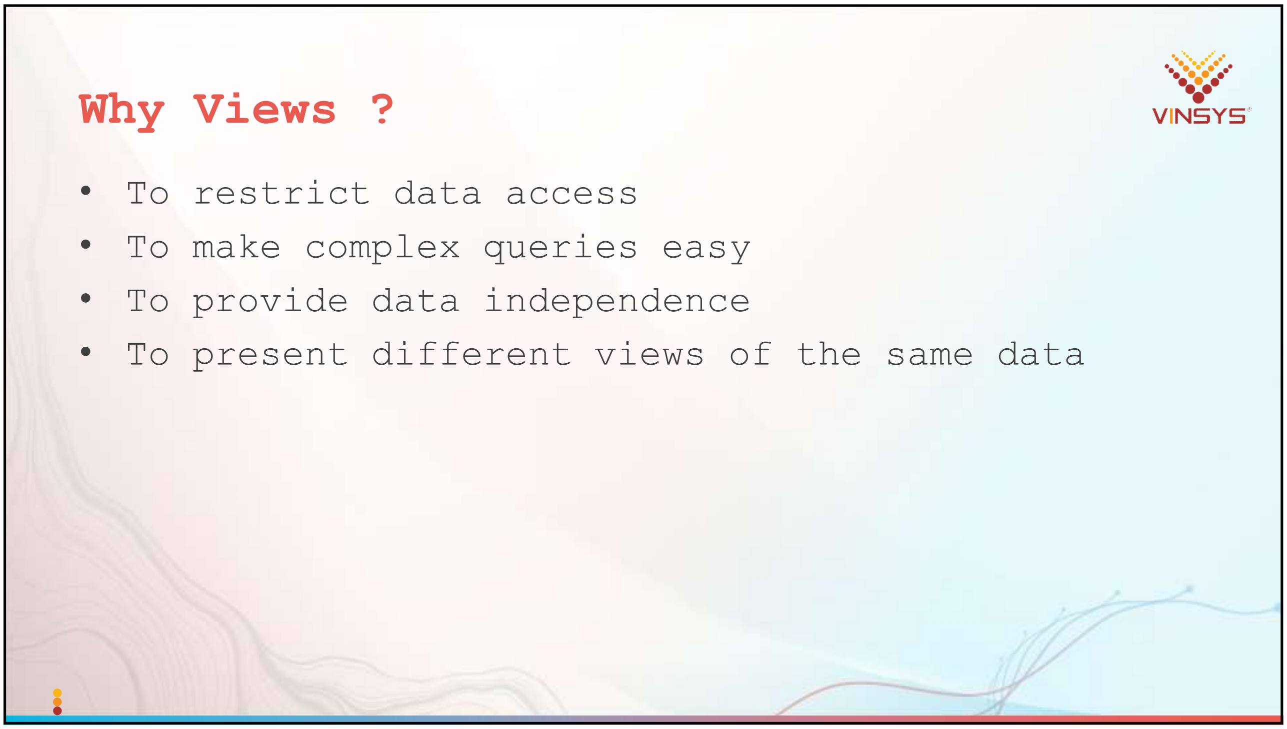
Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Numeric value generator
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

VIEW



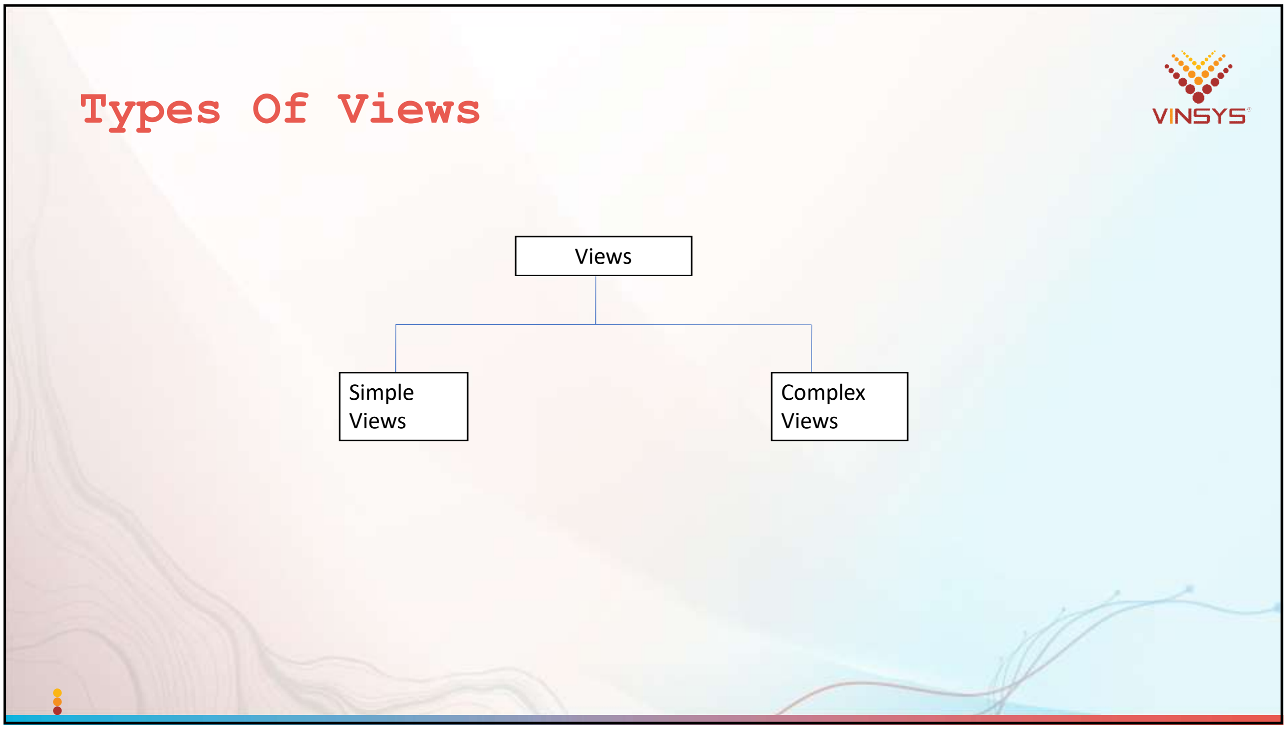
View

- A View is logical table based on a base table or a view in turn.
- A view contains no data of its own but is like a window through which data from tables can be viewed or changed.
- The view is stored as a SELECT statement in the data dictionary.



Why Views ?

- To restrict data access
- To make complex queries easy
- To provide data independence
- To present different views of the same data

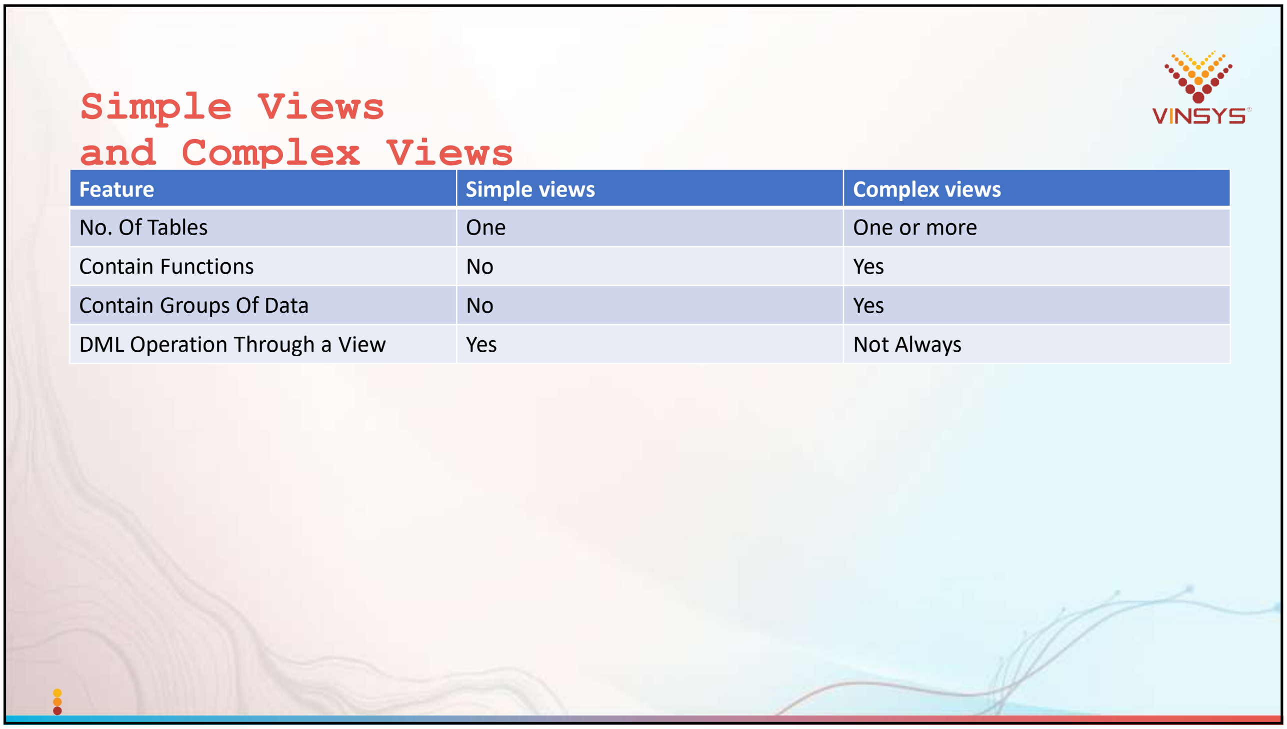


Types Of Views

Views

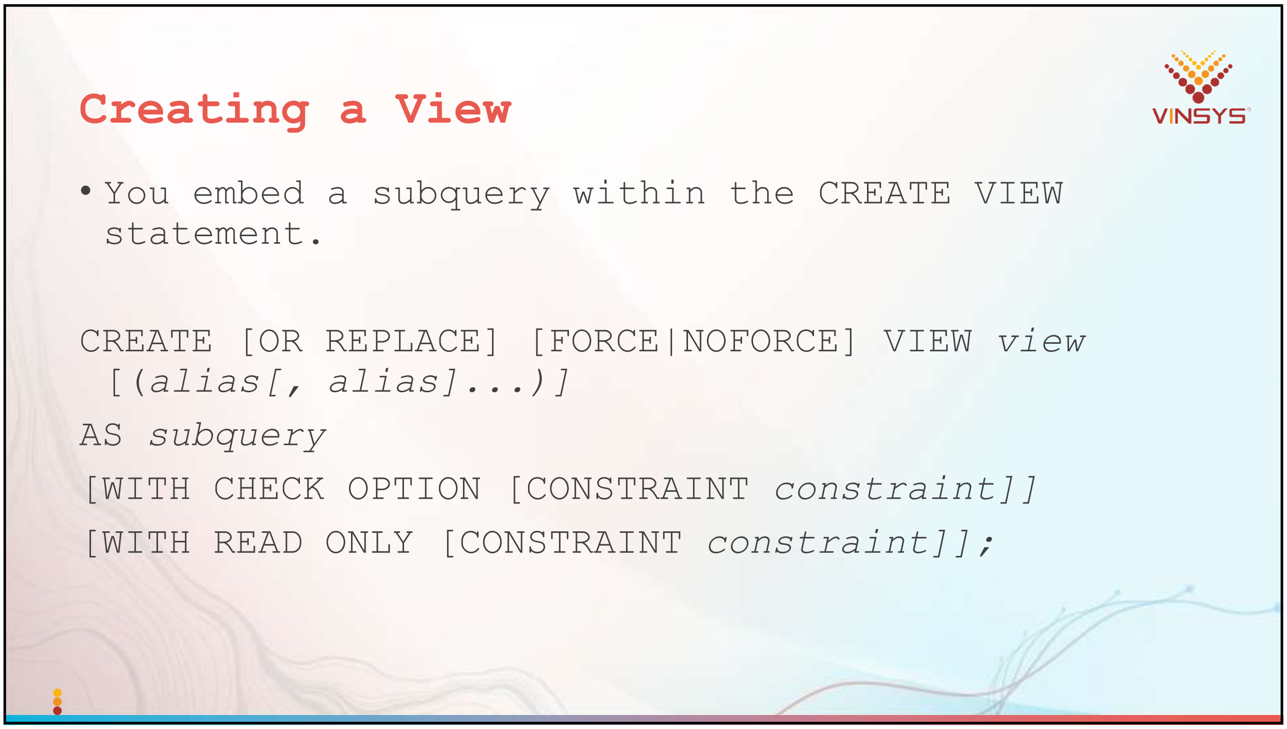
Simple Views

Complex Views



Simple Views and Complex Views

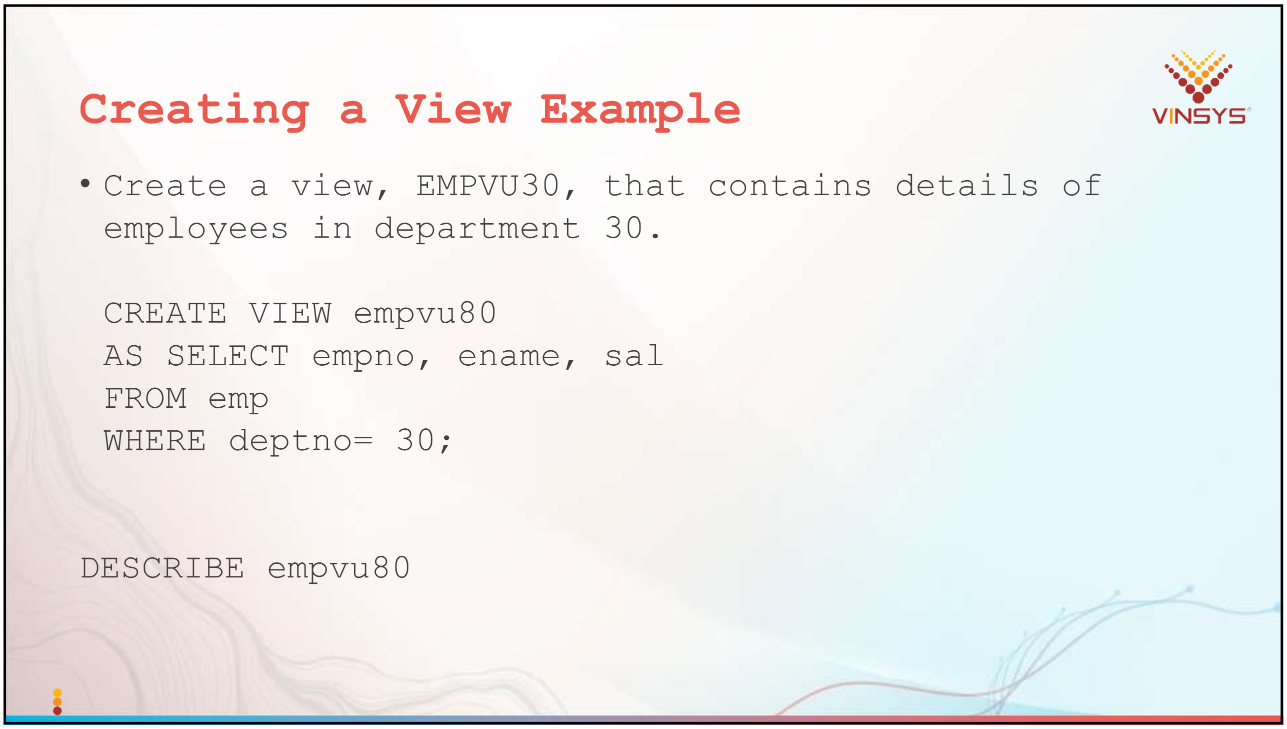
Feature	Simple views	Complex views
No. Of Tables	One	One or more
Contain Functions	No	Yes
Contain Groups Of Data	No	Yes
DML Operation Through a View	Yes	Not Always



Creating a View

- You embed a subquery within the CREATE VIEW statement.

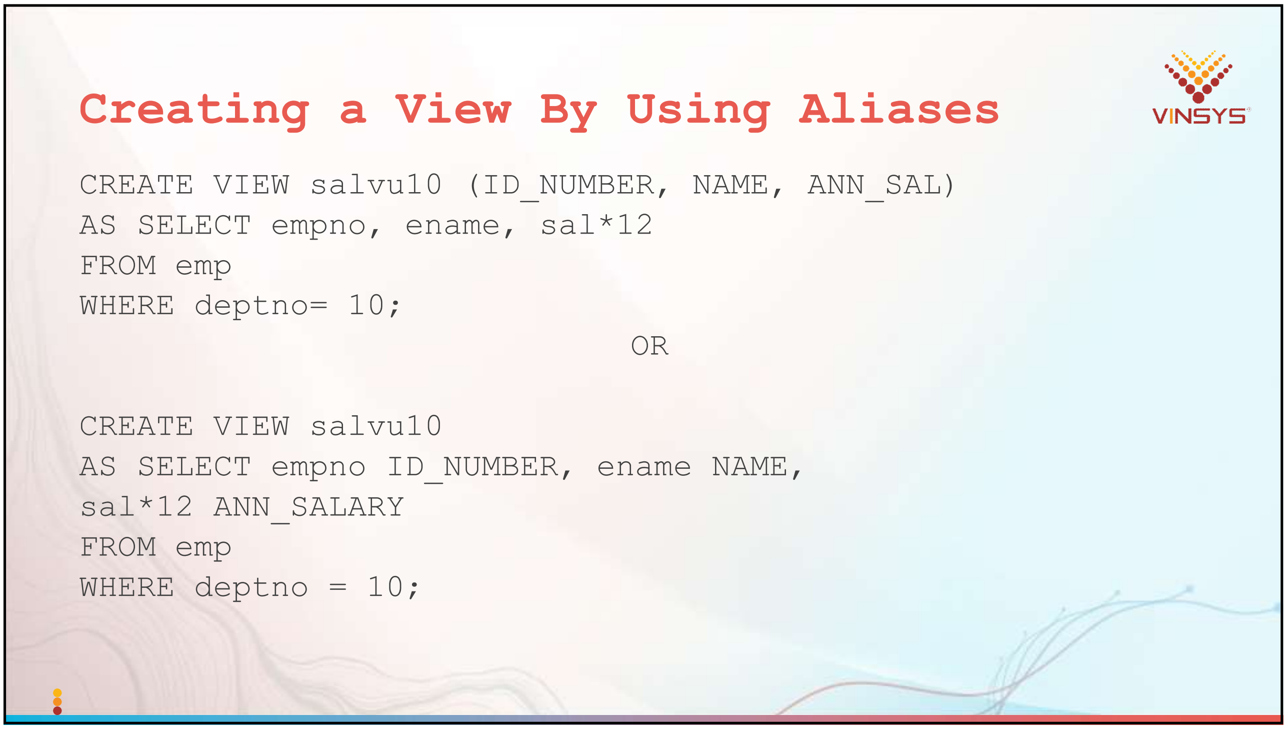
```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
  [(alias[, alias]...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```



Creating a View Example

- Create a view, EMPVU30, that contains details of employees in department 30.

```
CREATE VIEW empvu80
AS SELECT empno, ename, sal
FROM emp
WHERE deptno= 30;


DESCRIBE empvu80
```



Creating a View By Using Aliases

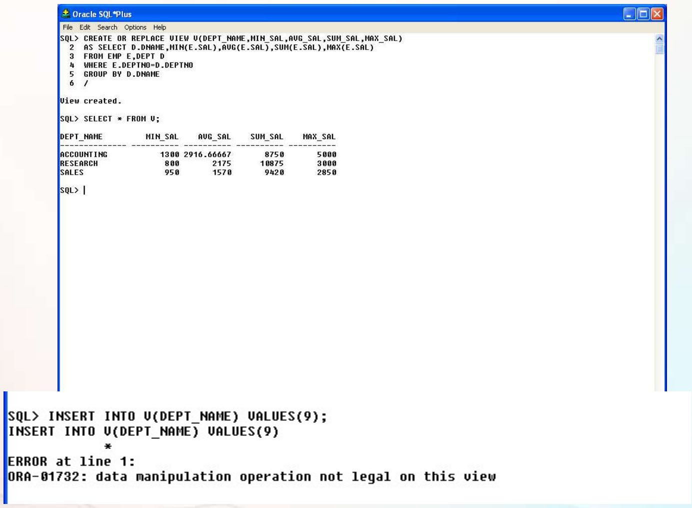
```
CREATE VIEW salvu10 (ID_NUMBER, NAME, ANN_SAL)
AS SELECT empno, ename, sal*12
FROM emp
WHERE deptno= 10;
                             OR

CREATE VIEW salvu10
AS SELECT empno ID_NUMBER, ename NAME,
sal*12 ANN_SALARY
FROM emp
WHERE deptno = 10;
```

Modifying a View

- There is no 'ALTER VIEW' command to modify a view.

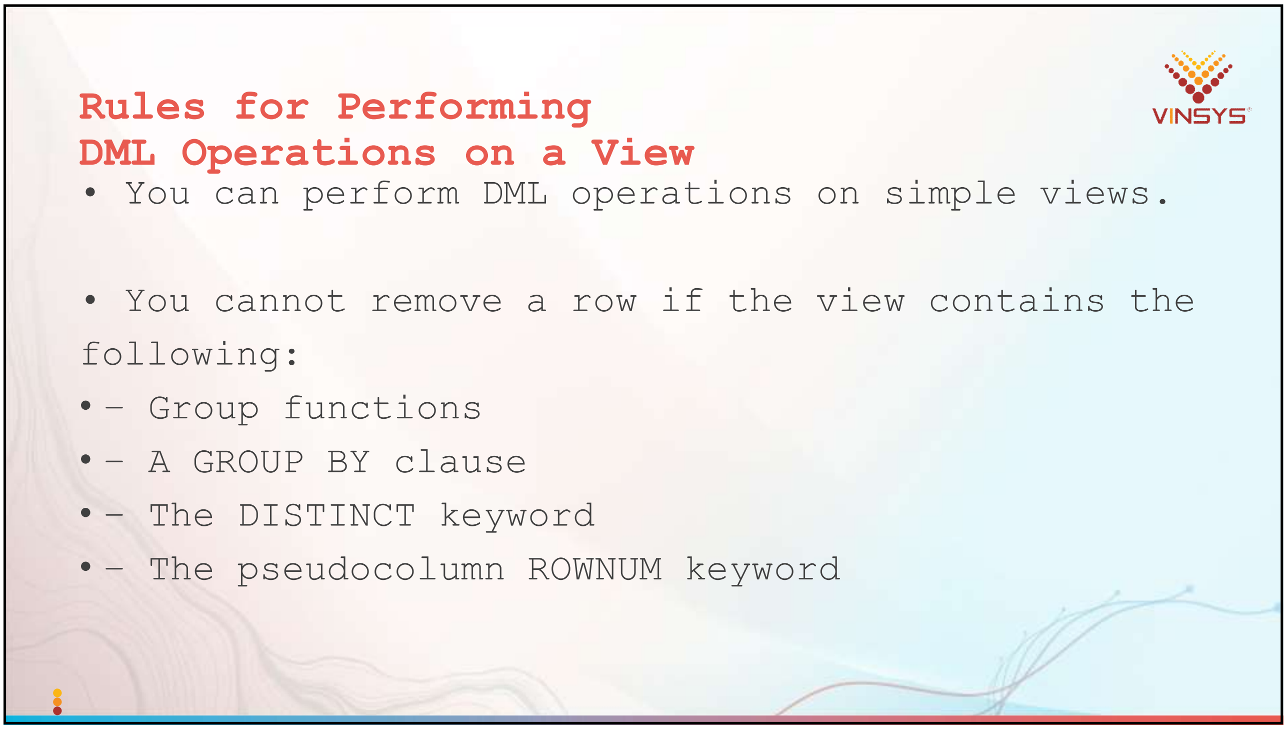
You have to use 'CREATE OR REPLACE VIEW *view*'

Creating Complex Views



```
SQL> CREATE OR REPLACE VIEW V(DEPT_NAME,MIN_SAL,AVG_SAL,SUM_SAL,MAX_SAL)
     AS SELECT D.DNAME,MIN(E.SAL),AVG(E.SAL),SUM(E.SAL),MAX(E.SAL)
     FROM EMP E,DEPT D
     WHERE E.DEPTNO=D.DEPTNO
     GROUP BY D.DNAME
     /

View created.

SQL> SELECT * FROM V;

DEPT_NAME         MIN_SAL    AVG_SAL    SUM_SAL    MAX_SAL
-------------- ---------- ---------- ---------- ----------
ACCOUNTING           1300 2916.66667       8750       5000
RESEARCH              800       2175      10875       3000
SALES                 950       1570       9420       2850

SQL> |
```

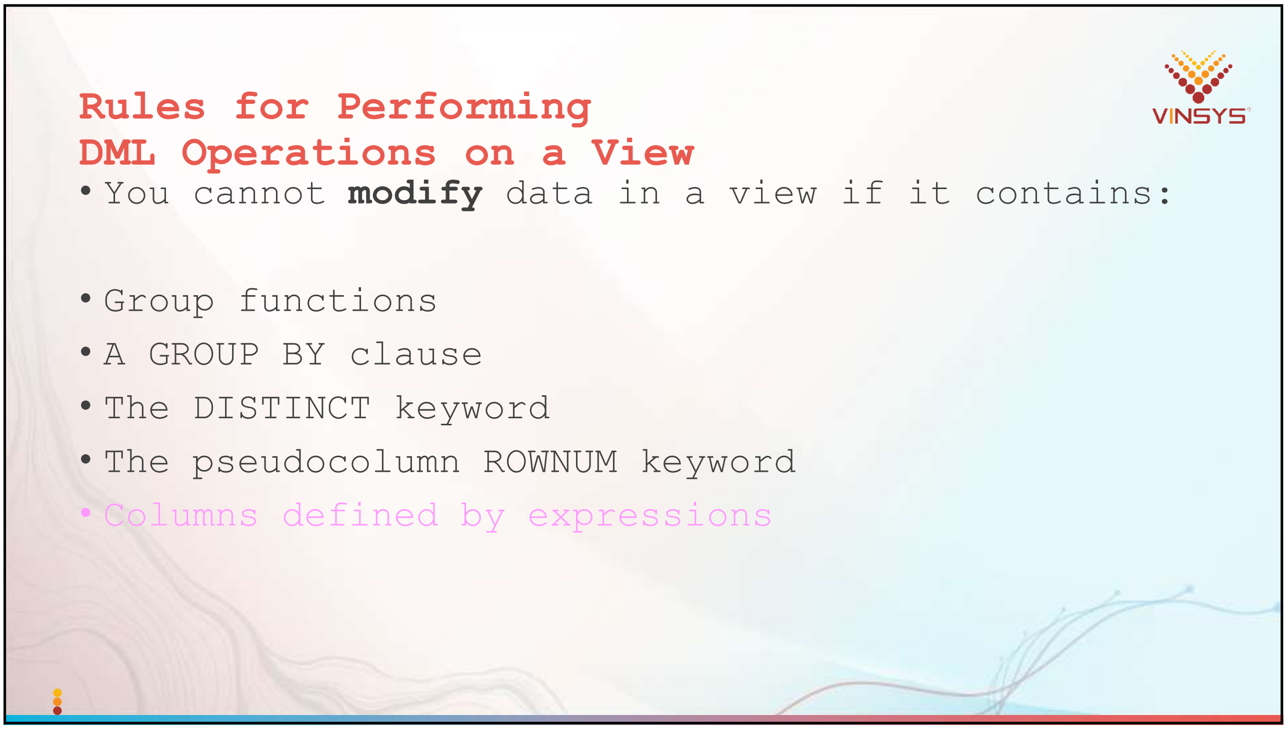
```
SQL> INSERT INTO V(DEPT_NAME) VALUES(9);
INSERT INTO V(DEPT_NAME) VALUES(9)
            *
ERROR at line 1:
ORA-01732: data manipulation operation not legal on this view
```



Rules for Performing DML Operations on a View

- You can perform DML operations on simple views.

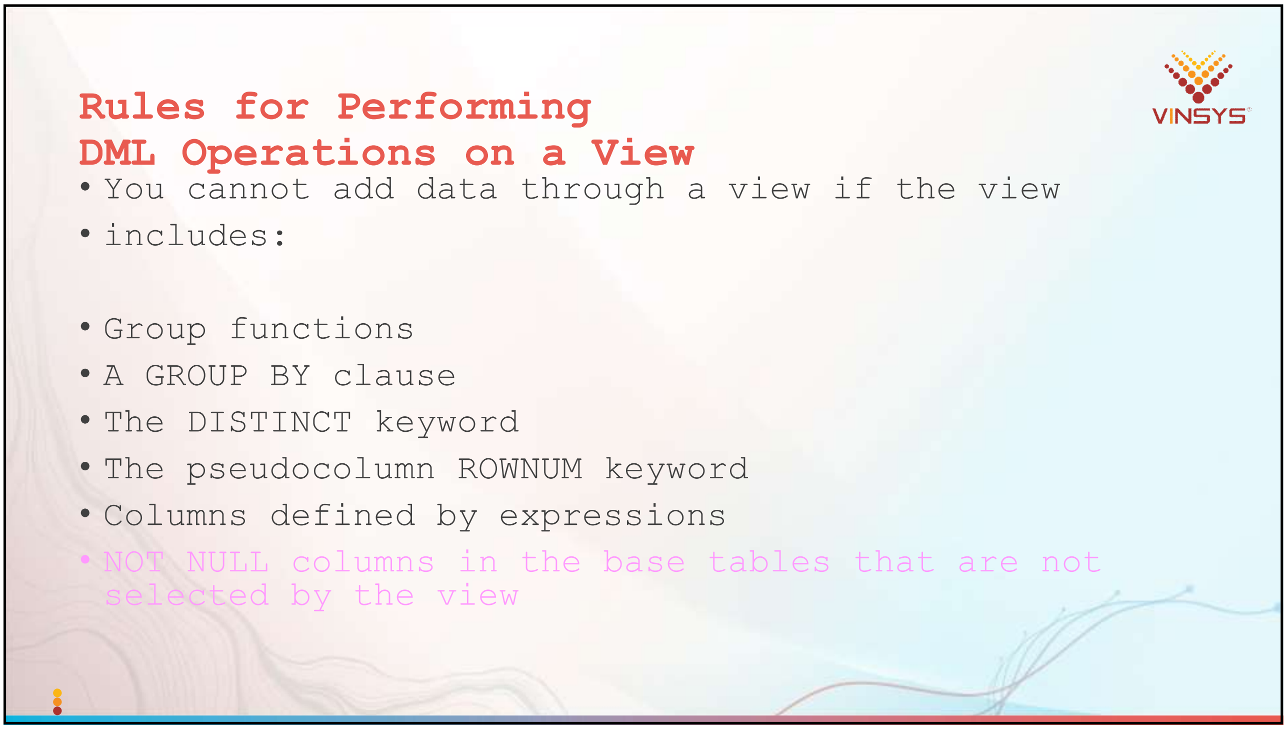
- You cannot remove a row if the view contains the following:
- – Group functions
- – A GROUP BY clause
- – The DISTINCT keyword
- – The pseudocolumn ROWNUM keyword



Rules for Performing DML Operations on a View

- You cannot **modify** data in a view if it contains:

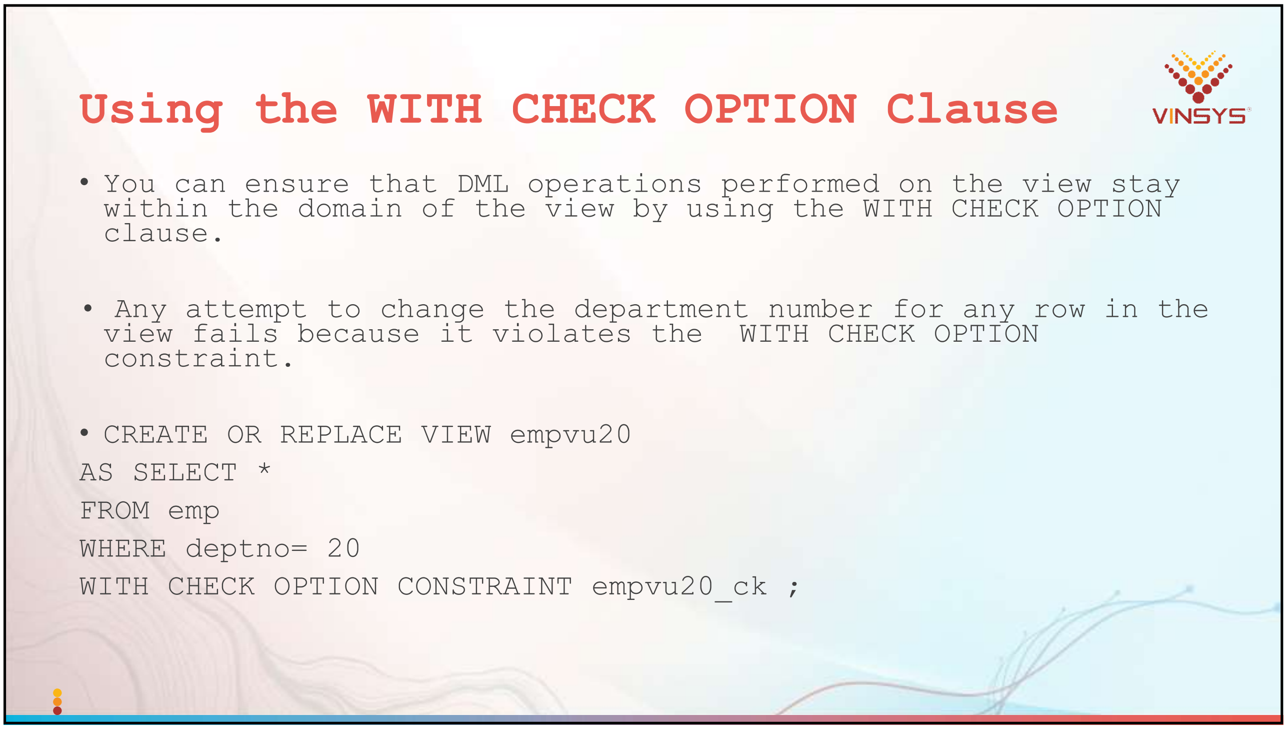
- Group functions
- A GROUP BY clause
- The DISTINCT keyword
- The pseudocolumn ROWNUM keyword
- Columns defined by expressions



Rules for Performing DML Operations on a View

- You cannot add data through a view if the view
- includes:

- Group functions
- A GROUP BY clause
- The DISTINCT keyword
- The pseudocolumn ROWNUM keyword
- Columns defined by expressions
- NOT NULL columns in the base tables that are not selected by the view



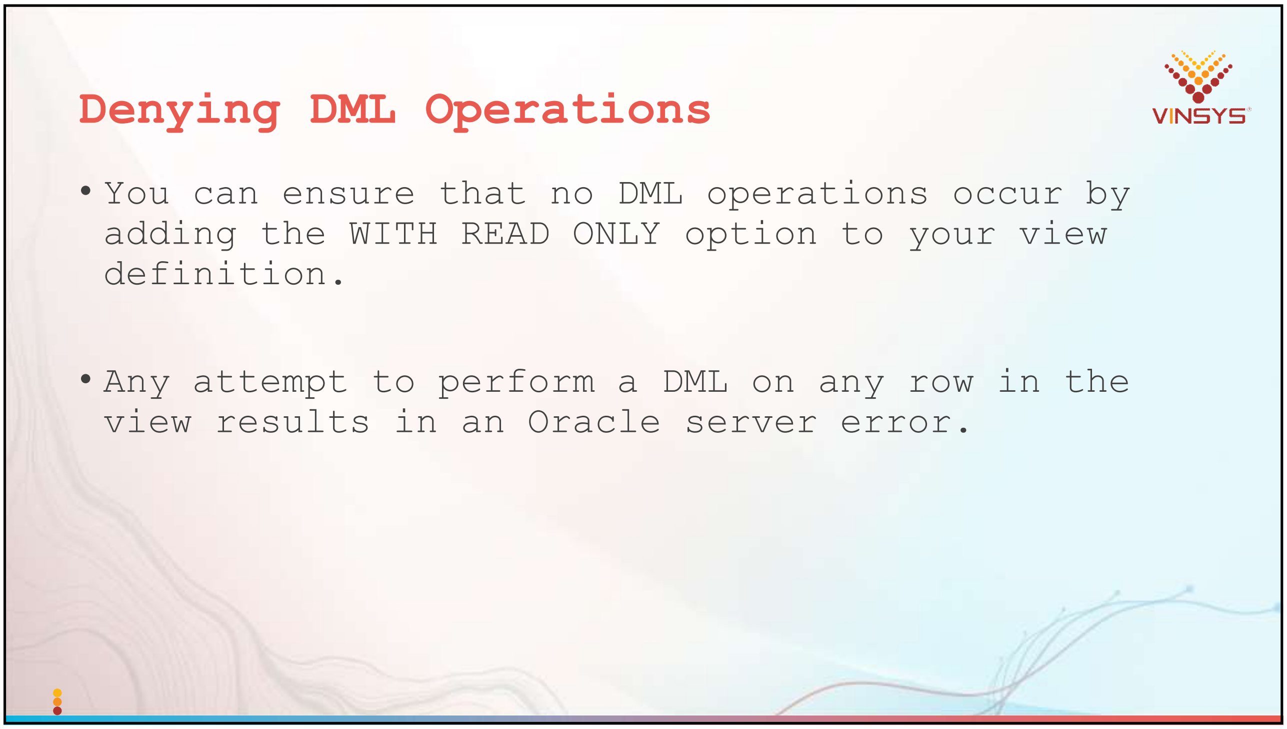
Using the WITH CHECK OPTION Clause

- You can ensure that DML operations performed on the view stay within the domain of the view by using the WITH CHECK OPTION clause.

- Any attempt to change the department number for any row in the view fails because it violates the WITH CHECK OPTION constraint.

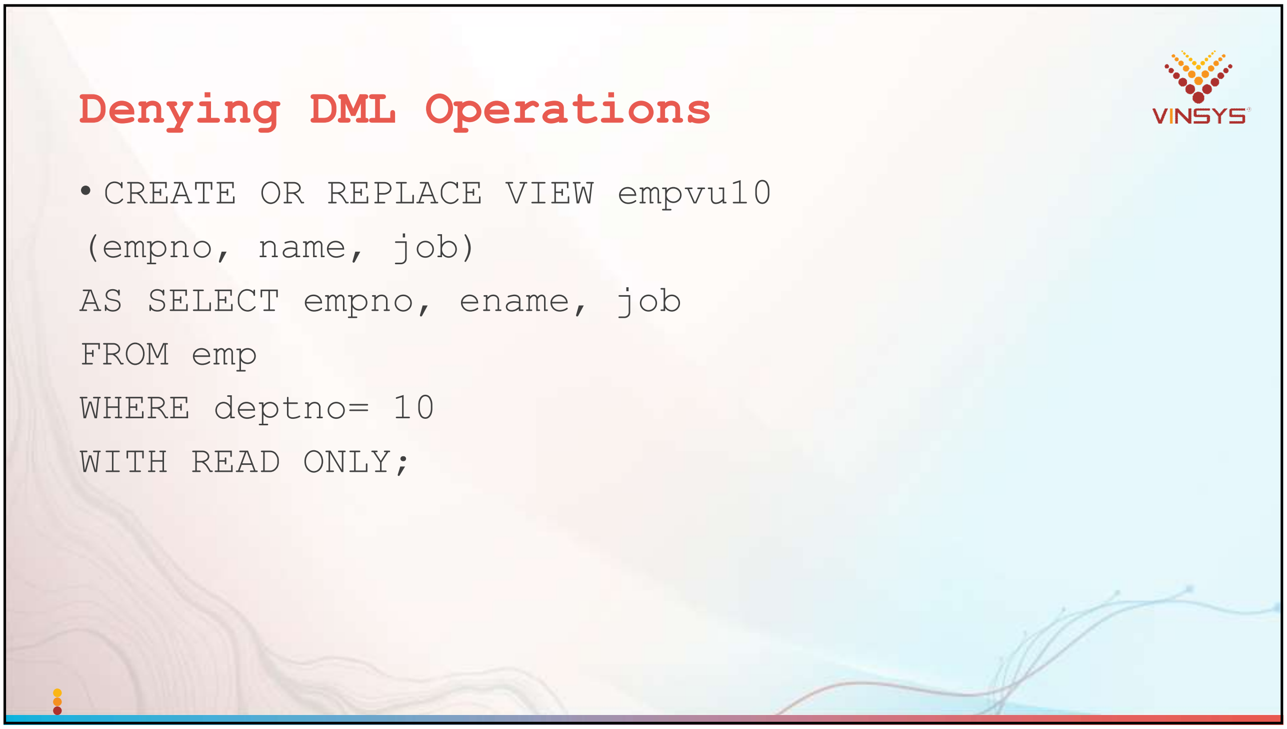
- CREATE OR REPLACE VIEW empvu20

```
AS SELECT *
FROM emp
WHERE deptno= 20
WITH CHECK OPTION CONSTRAINT empvu20_ck ;
```



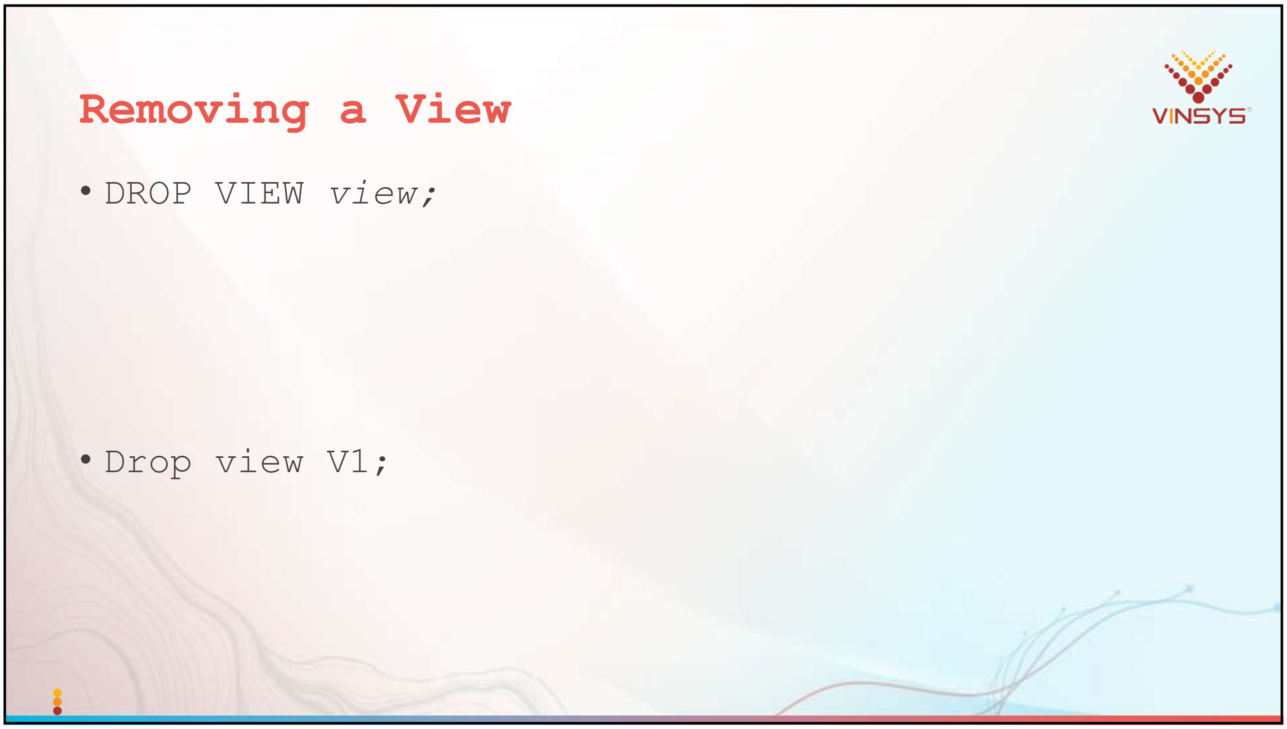
Denying DML Operations

- You can ensure that no DML operations occur by adding the WITH READ ONLY option to your view definition.
- Any attempt to perform a DML on any row in the view results in an Oracle server error.



Denying DML Operations

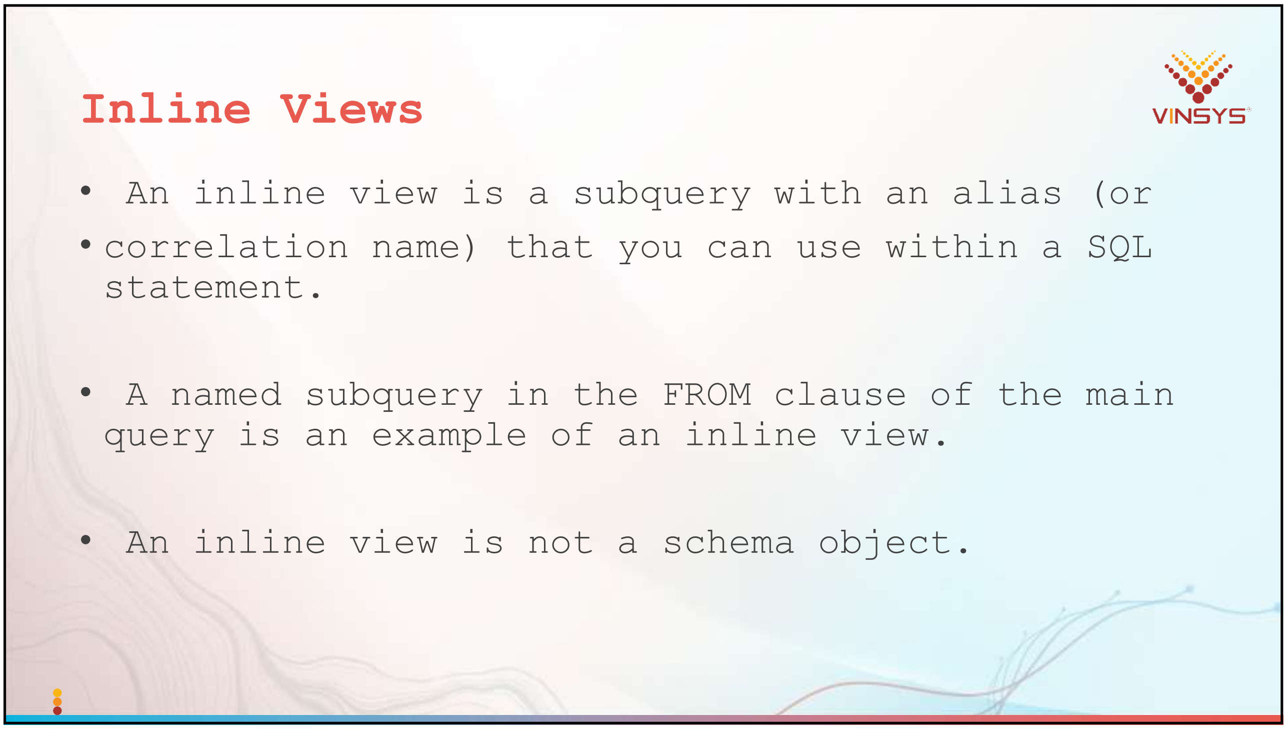
```
• CREATE OR REPLACE VIEW empvu10
(empno, name, job)
AS SELECT empno, ename, job
FROM emp
WHERE deptno= 10
WITH READ ONLY;
```



Removing a View

- DROP VIEW *view;*

- Drop view V1;

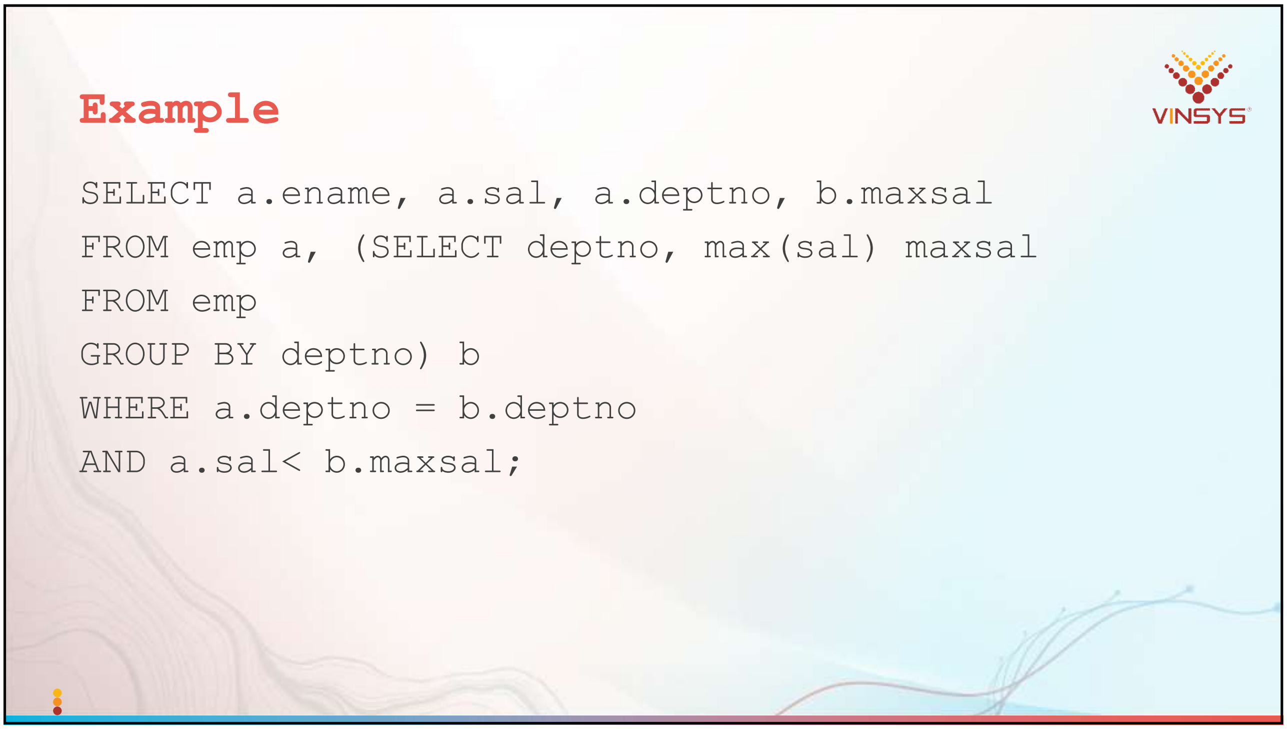


Inline Views

- An inline view is a subquery with an alias (or
- correlation name) that you can use within a SQL statement.

- A named subquery in the FROM clause of the main query is an example of an inline view.

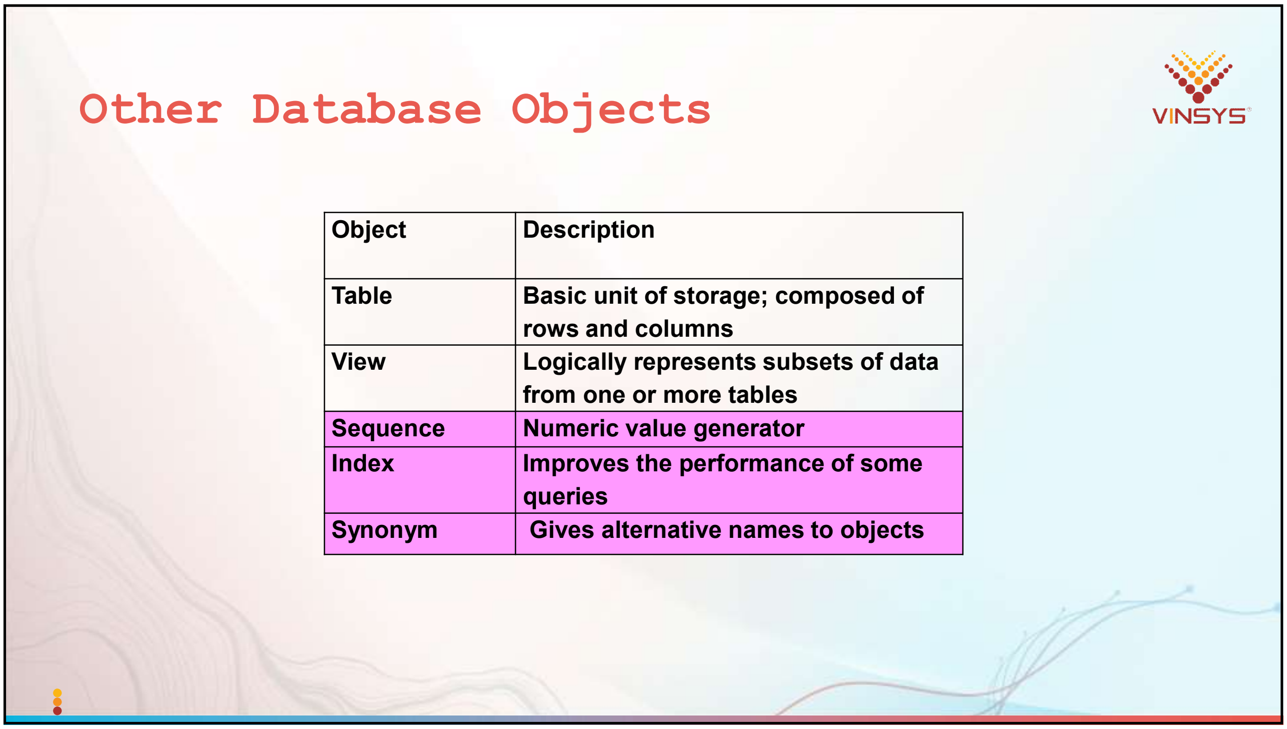
- An inline view is not a schema object.



Example

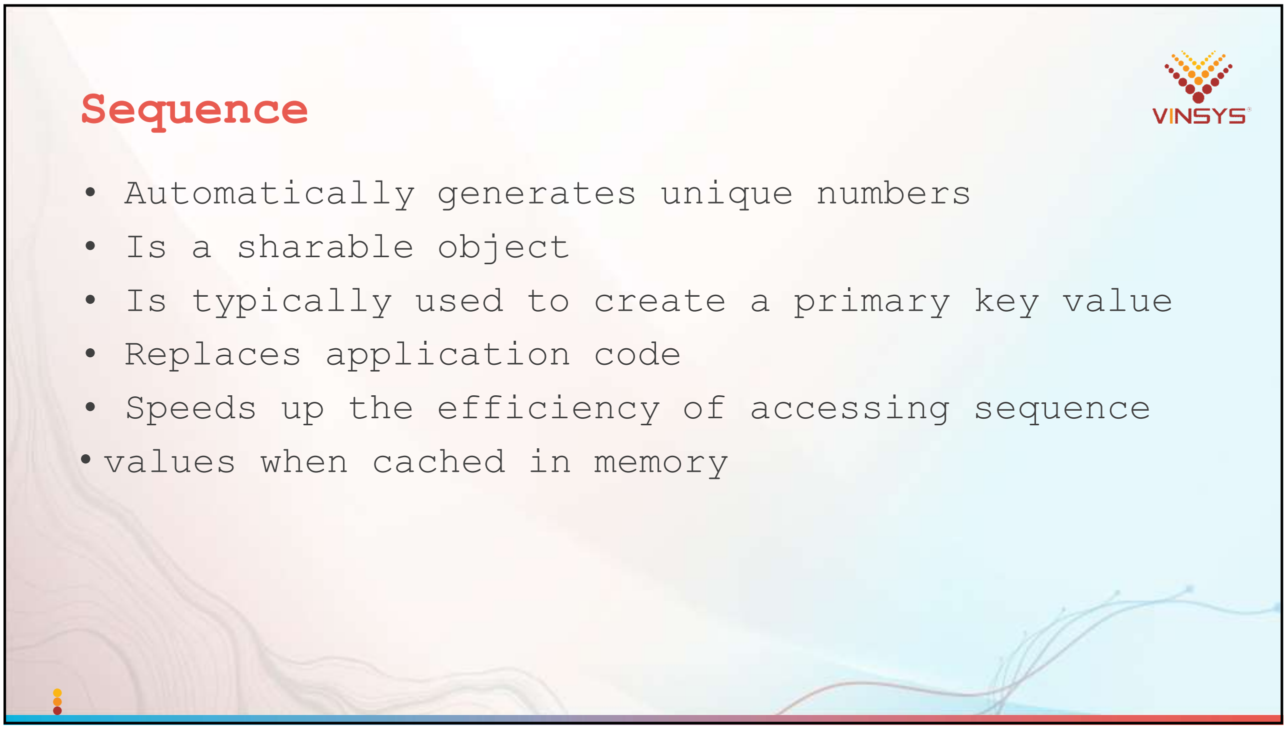
```
SELECT a.ename, a.sal, a.deptno, b.maxsal
FROM emp a, (SELECT deptno, max(sal) maxsal
FROM emp
GROUP BY deptno) b
WHERE a.deptno = b.deptno
AND a.sal< b.maxsal;
```

Other Database Objects



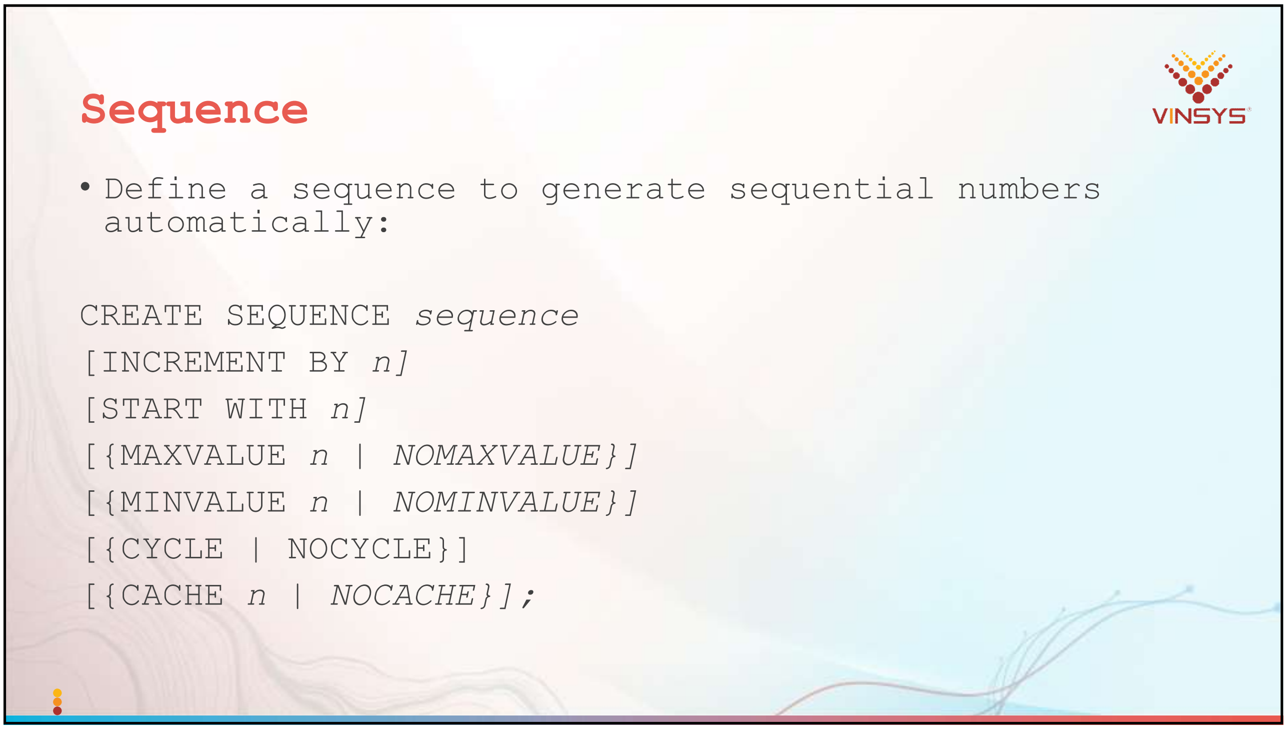
Other Database Objects

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Numeric value generator
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects



Sequence

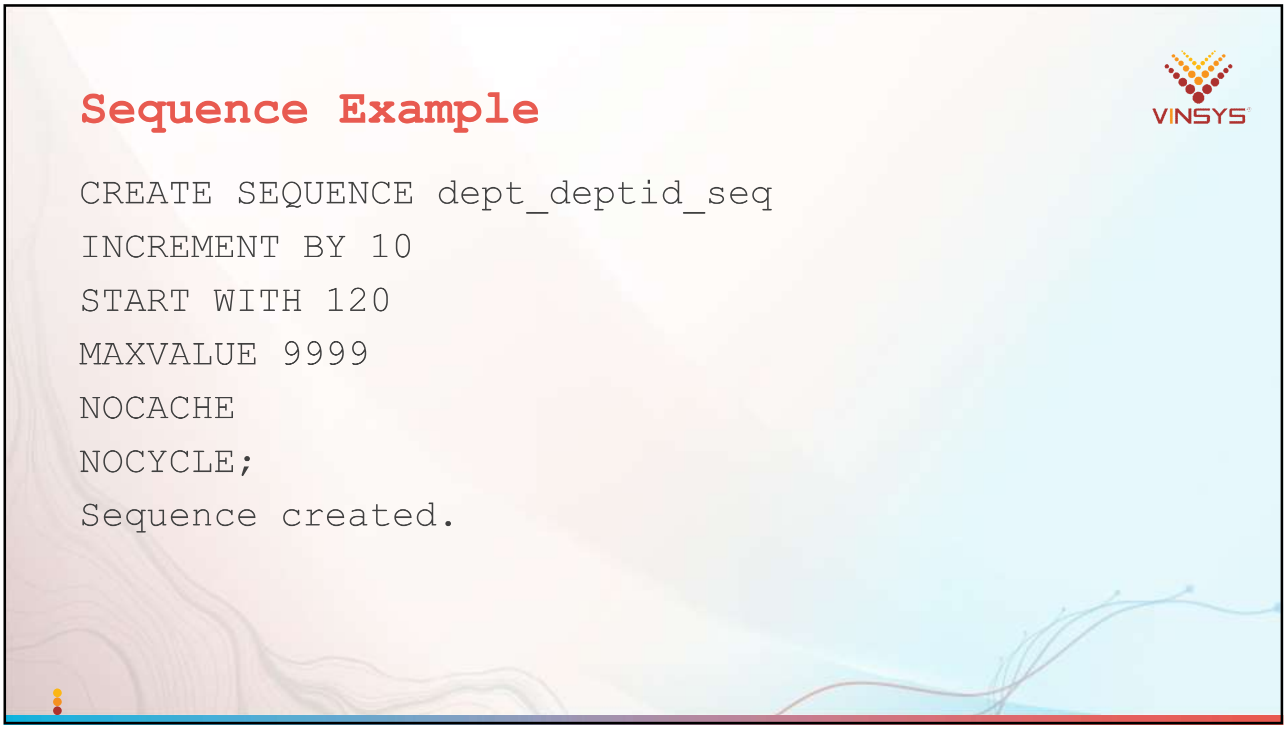
- Automatically generates unique numbers
- Is a sharable object
- Is typically used to create a primary key value
- Replaces application code
- Speeds up the efficiency of accessing sequence
- values when cached in memory



Sequence

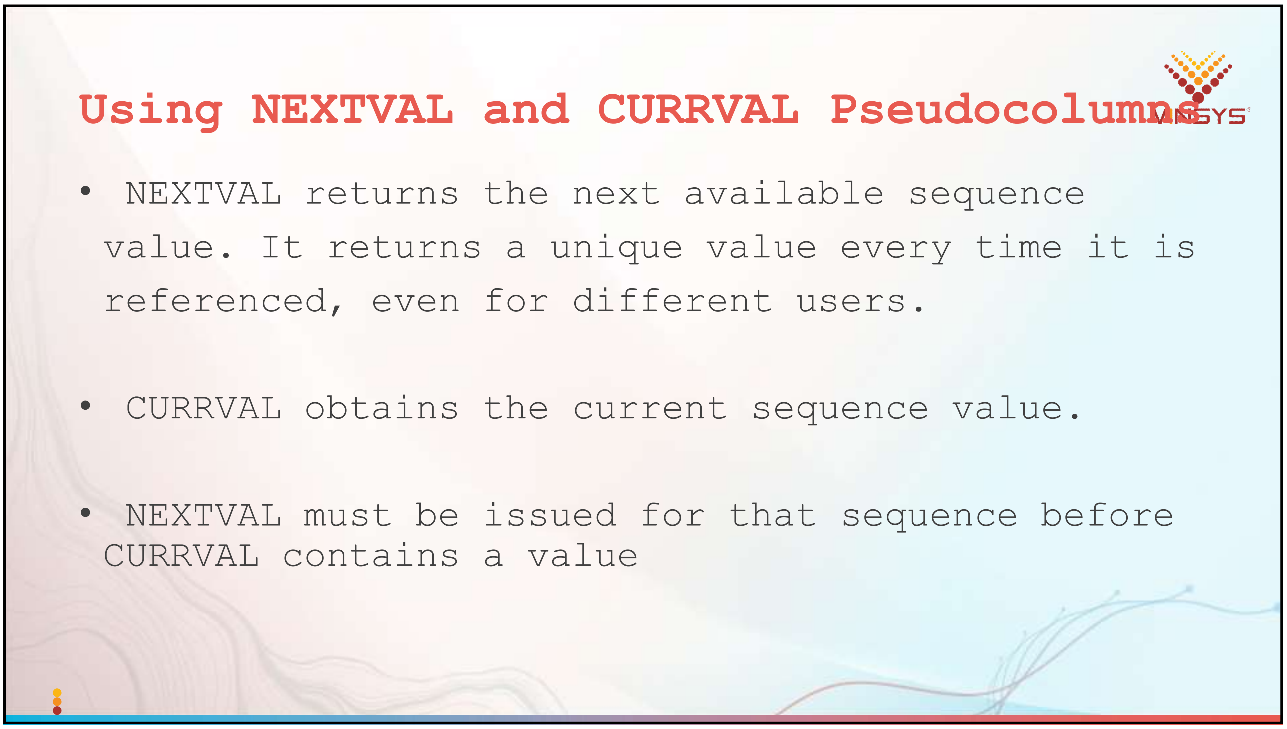
- Define a sequence to generate sequential numbers automatically:

```
CREATE SEQUENCE sequence
[INCREMENT BY n]
[START WITH n]
[{MAXVALUE n | NOMAXVALUE}]
[{MINVALUE n | NOMINVALUE}]
[{CYCLE | NOCYCLE}]
[{CACHE n | NOCACHE}];
```



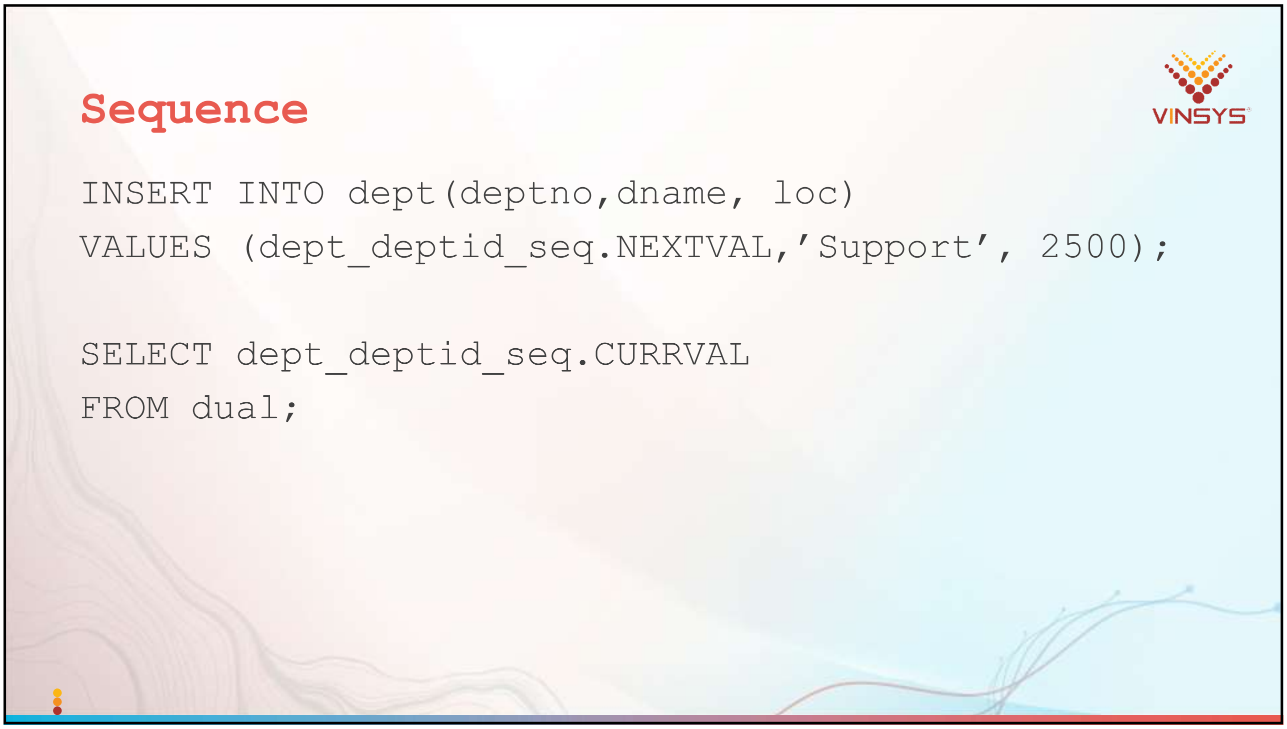
Sequence Example

```
CREATE SEQUENCE dept_deptid_seq
INCREMENT BY 10
START WITH 120
MAXVALUE 9999
NOCACHE
NOCYCLE;
Sequence created.
```



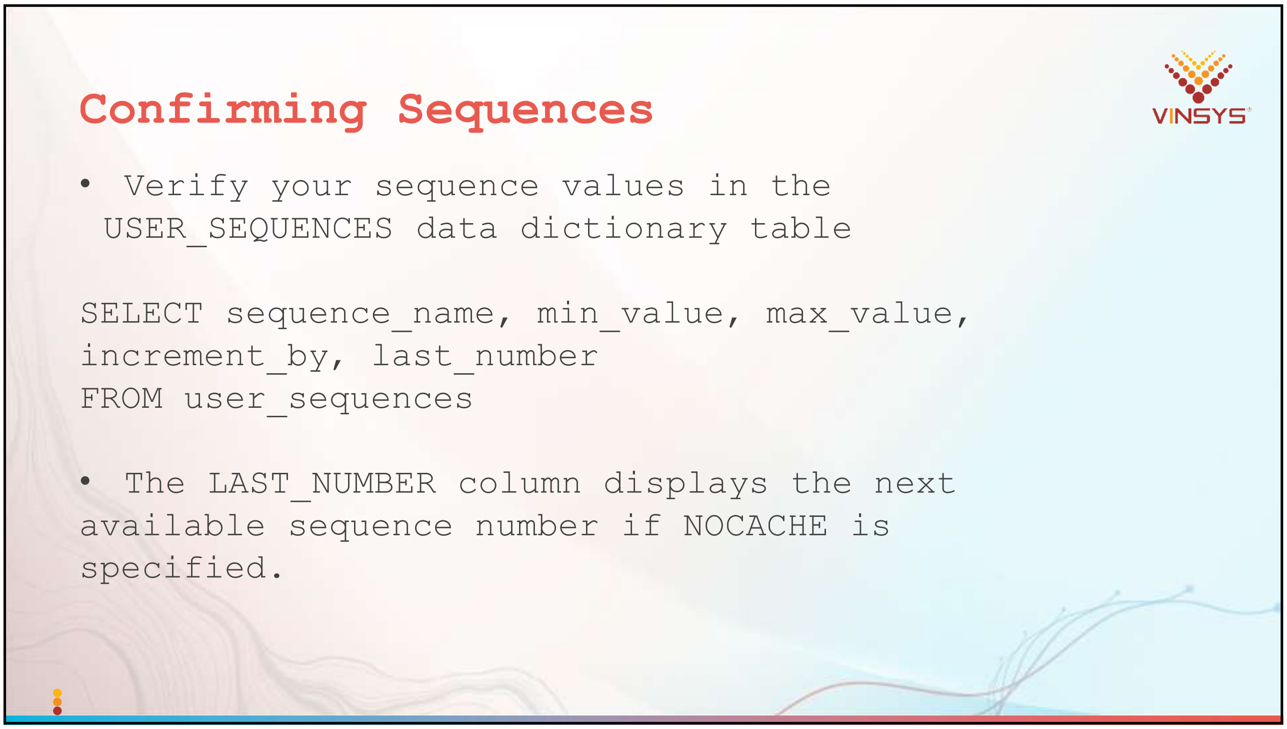
Using NEXTVAL and CURRVAL Pseudocolumns

- NEXTVAL returns the next available sequence value. It returns a unique value every time it is referenced, even for different users.
- CURRVAL obtains the current sequence value.
- NEXTVAL must be issued for that sequence before CURRVAL contains a value



Sequence

```
INSERT INTO dept(deptno,dname, loc)
VALUES (dept_deptid_seq.NEXTVAL,'Support', 2500);

SELECT dept_deptid_seq.CURRVAL
FROM dual;
```

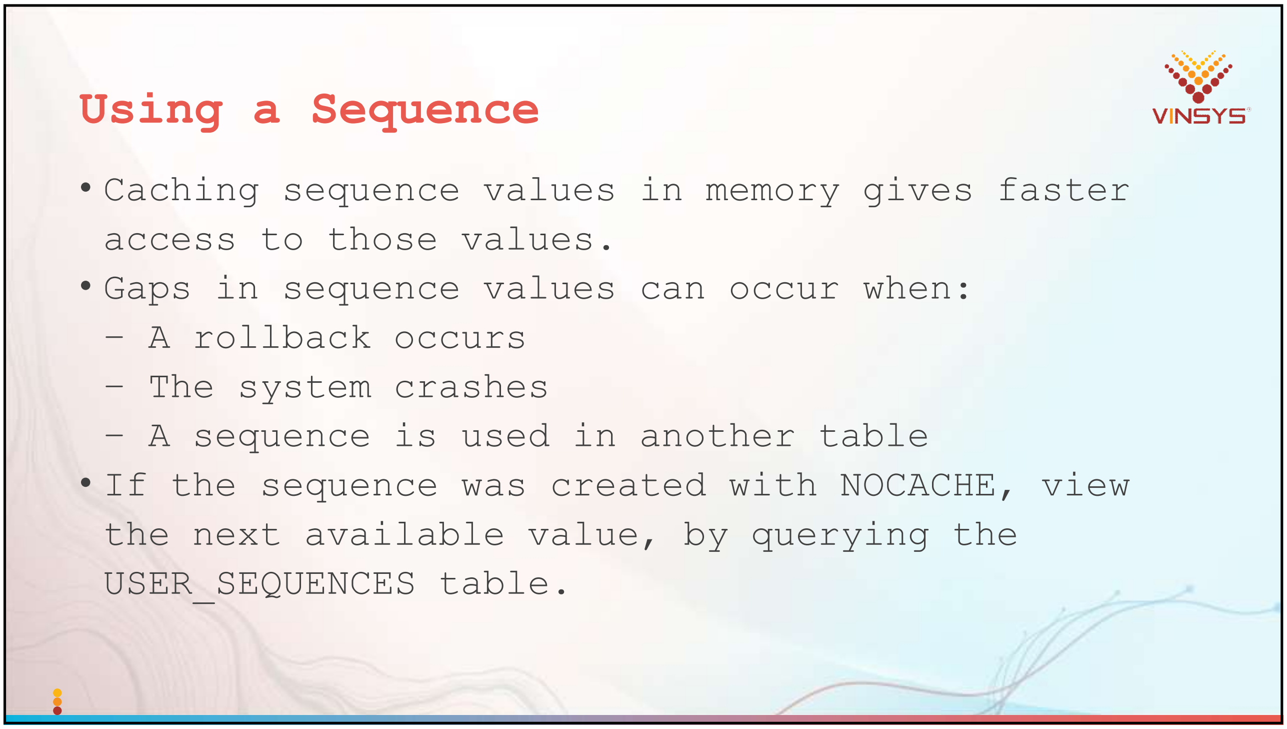


Confirming Sequences

- Verify your sequence values in the USER_SEQUENCES data dictionary table

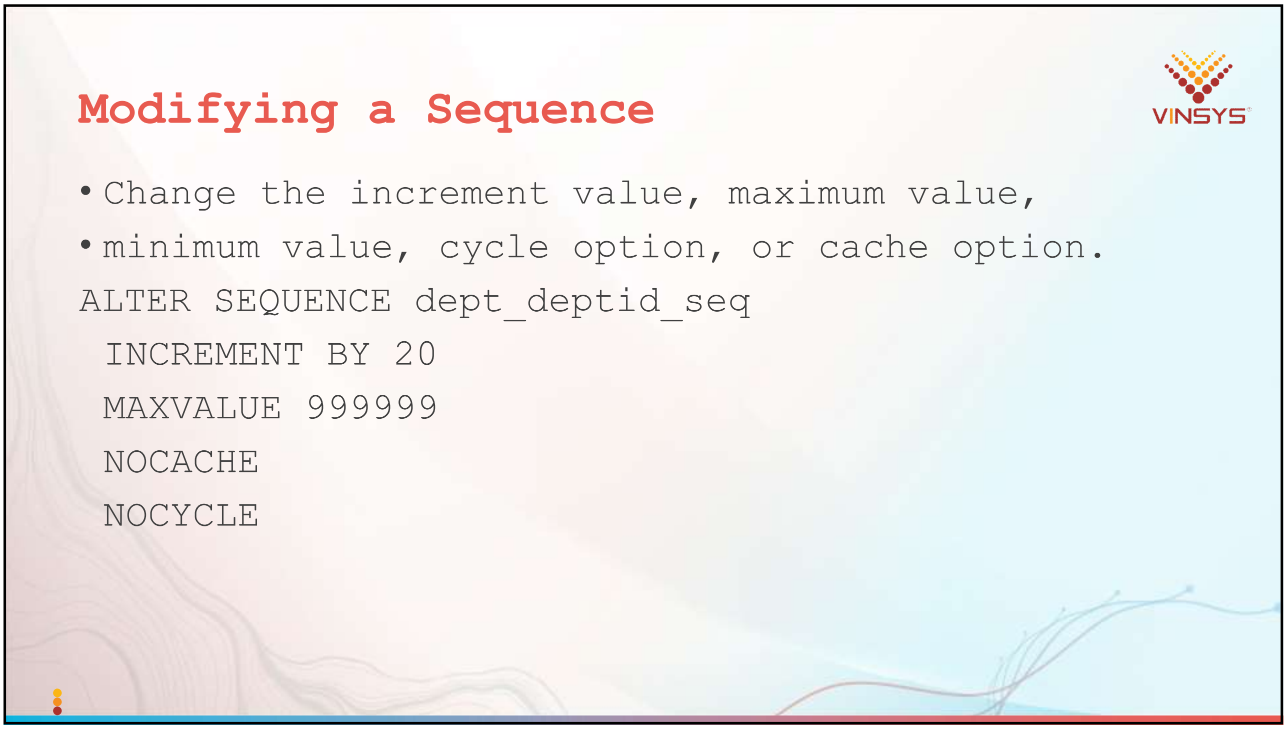
```
SELECT sequence_name, min_value, max_value,
increment_by, last_number
FROM user_sequences
```

- The LAST_NUMBER column displays the next available sequence number if NOCACHE is specified.



Using a Sequence

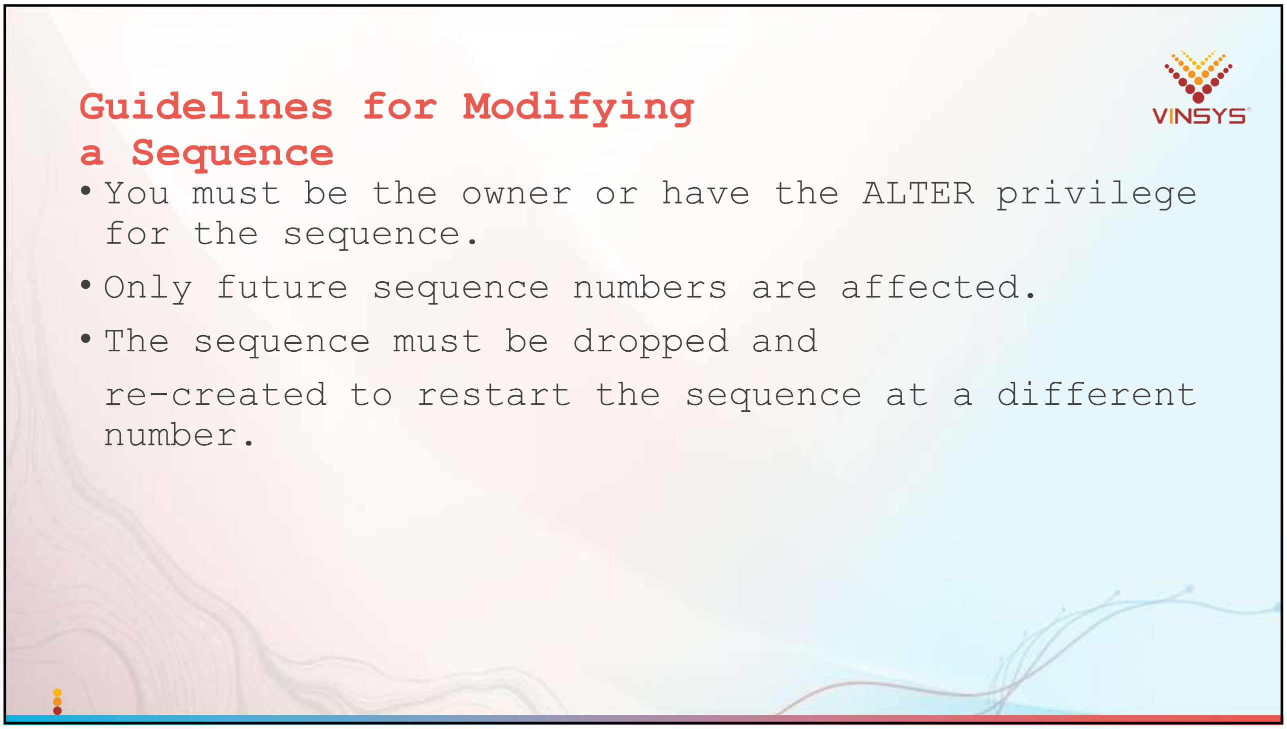
- Caching sequence values in memory gives faster access to those values.
- Gaps in sequence values can occur when:
  - A rollback occurs
  - The system crashes
  - A sequence is used in another table
- If the sequence was created with NOCACHE, view the next available value, by querying the USER_SEQUENCES table.



Modifying a Sequence

- Change the increment value, maximum value,
- minimum value, cycle option, or cache option.

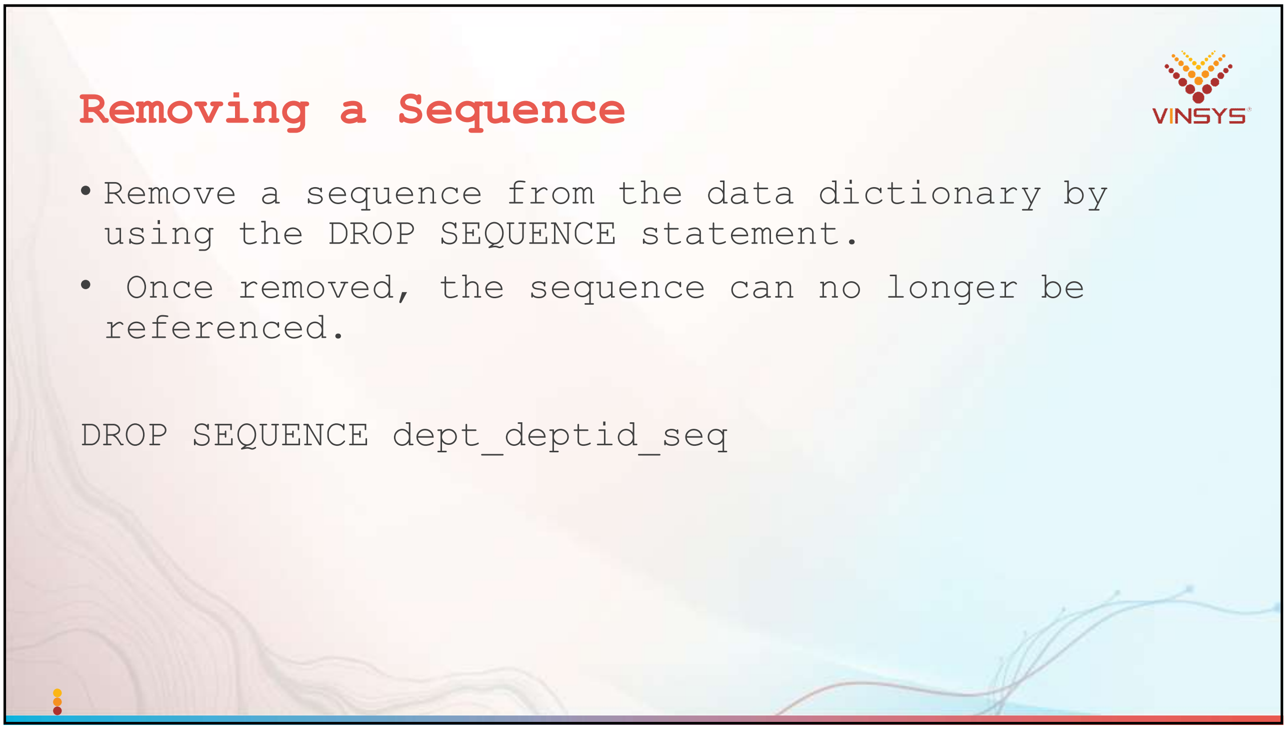
```
ALTER SEQUENCE dept_deptid_seq
 INCREMENT BY 20
 MAXVALUE 999999
 NOCACHE
 NOCYCLE
```



Guidelines for Modifying a Sequence

- You must be the owner or have the ALTER privilege for the sequence.
- Only future sequence numbers are affected.
- The sequence must be dropped and

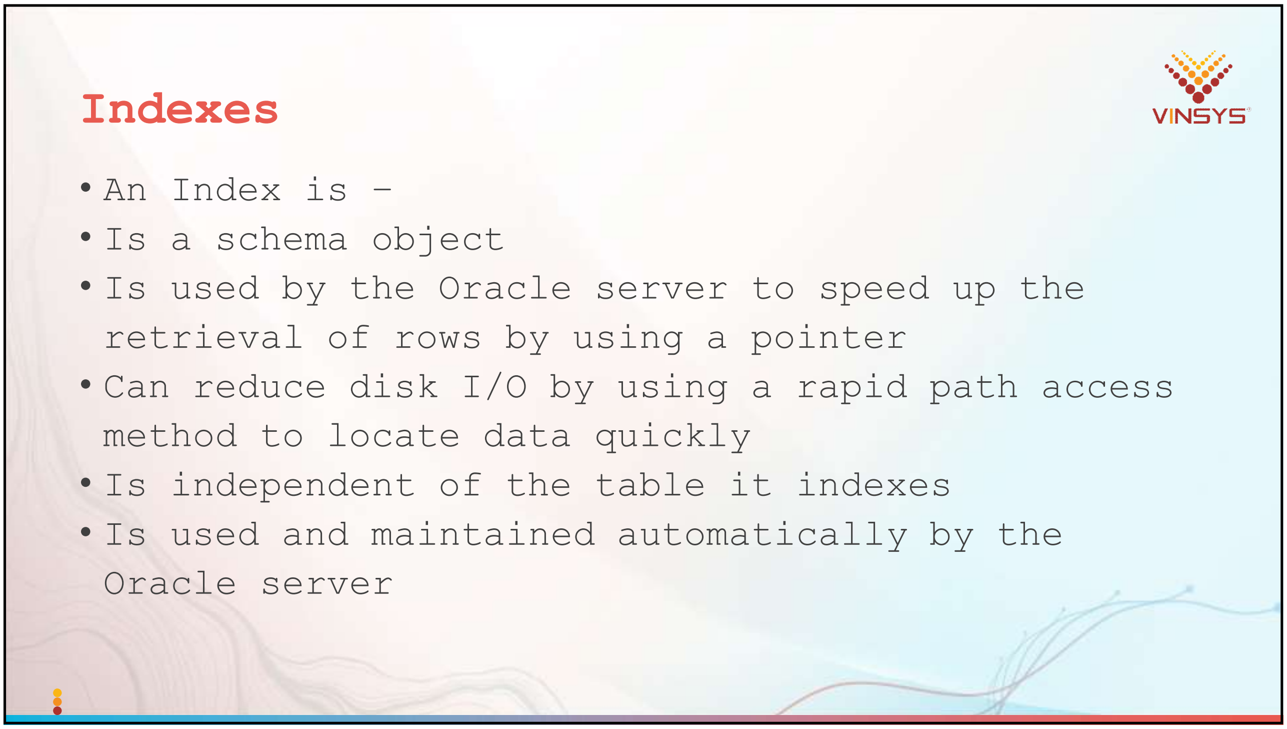
  re-created to restart the sequence at a different number.



Removing a Sequence

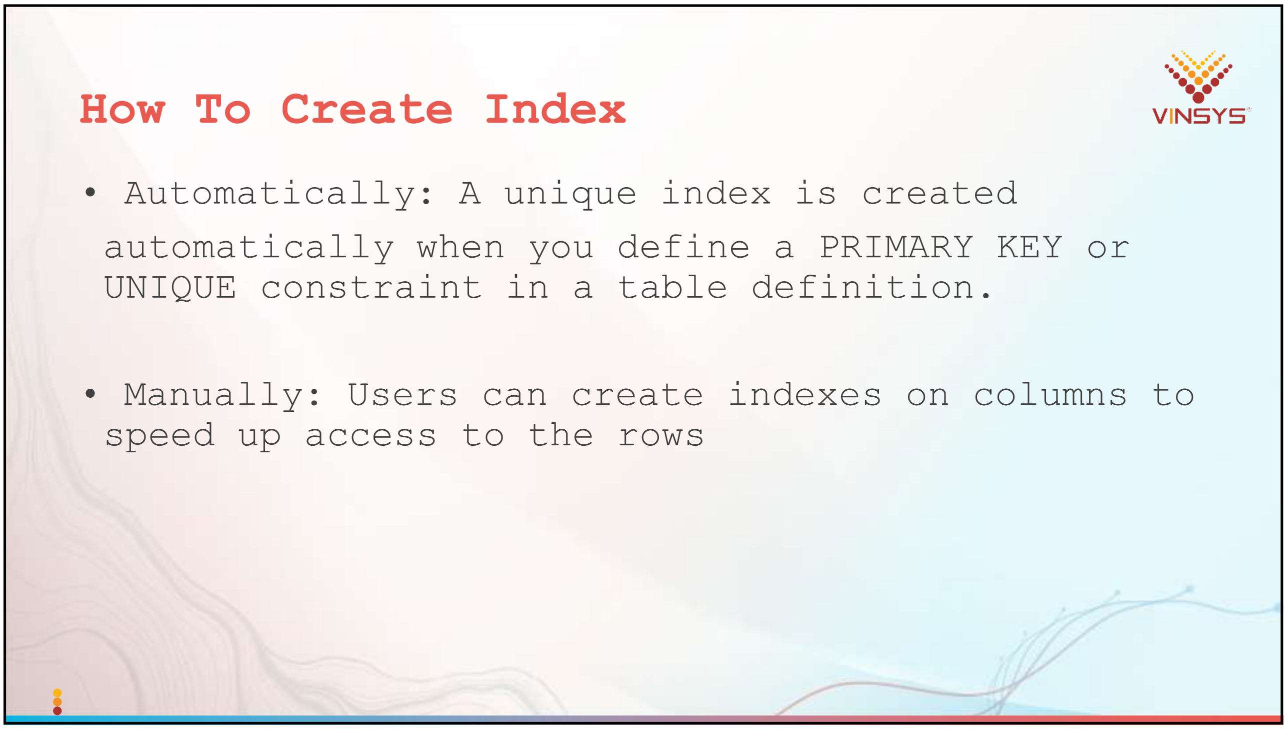
- Remove a sequence from the data dictionary by using the DROP SEQUENCE statement.
- Once removed, the sequence can no longer be referenced.

```
DROP SEQUENCE dept_deptid_seq
```



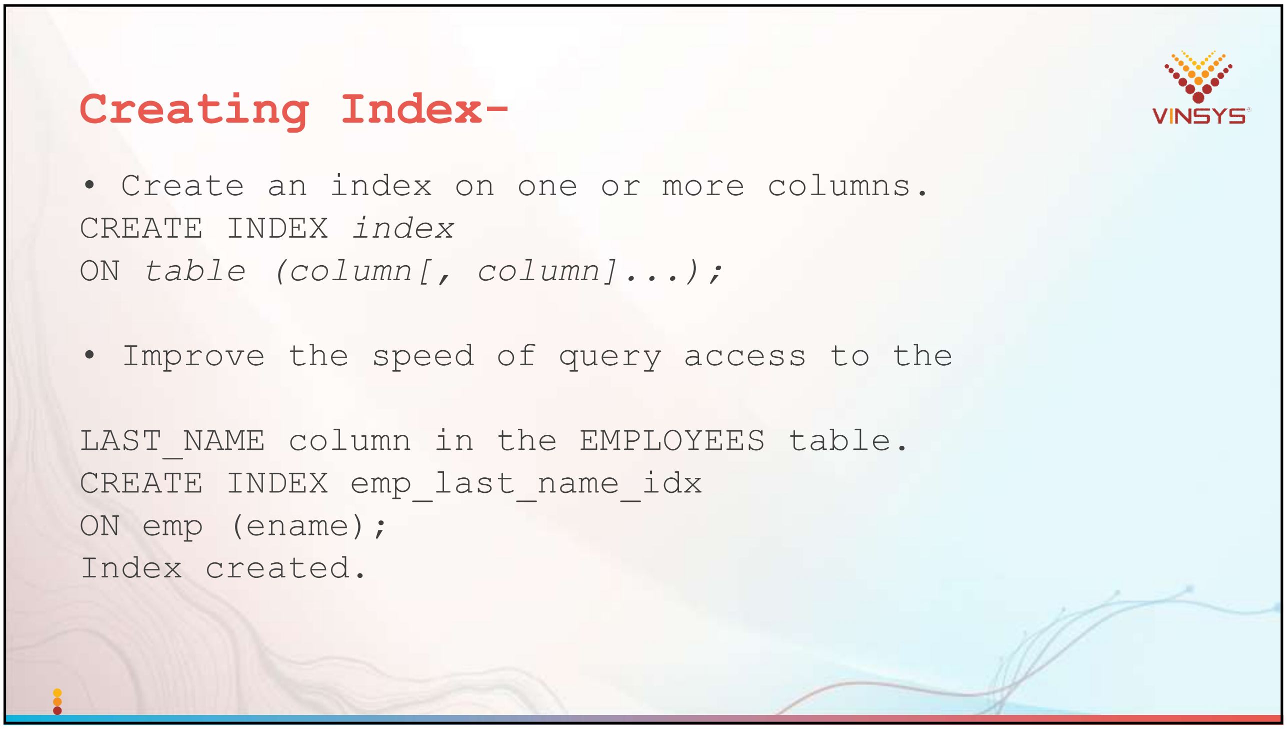
Indexes

- An Index is –
- Is a schema object
- Is used by the Oracle server to speed up the retrieval of rows by using a pointer
- Can reduce disk I/O by using a rapid path access method to locate data quickly
- Is independent of the table it indexes
- Is used and maintained automatically by the Oracle server



How To Create Index

- Automatically: A unique index is created automatically when you define a PRIMARY KEY or UNIQUE constraint in a table definition.
- Manually: Users can create indexes on columns to speed up access to the rows



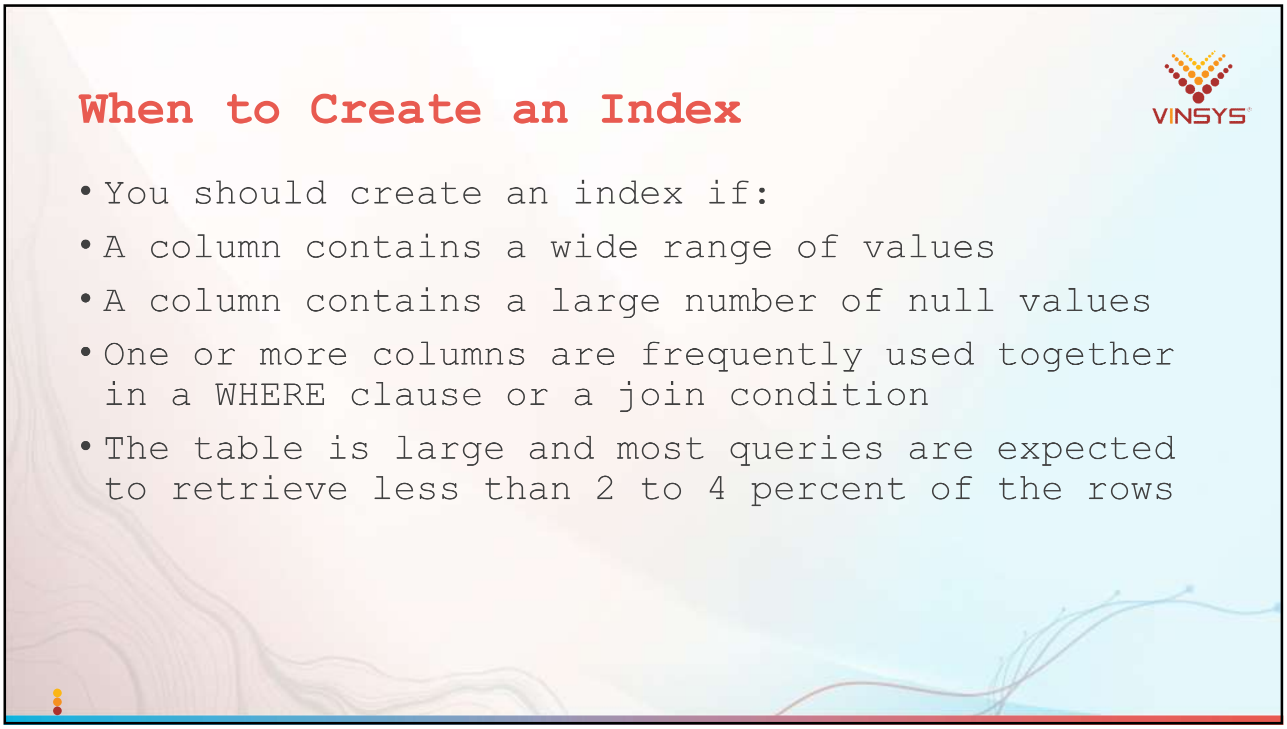
Creating Index-

- Create an index on one or more columns.

```
CREATE INDEX index
ON table (column[, column]...);
```

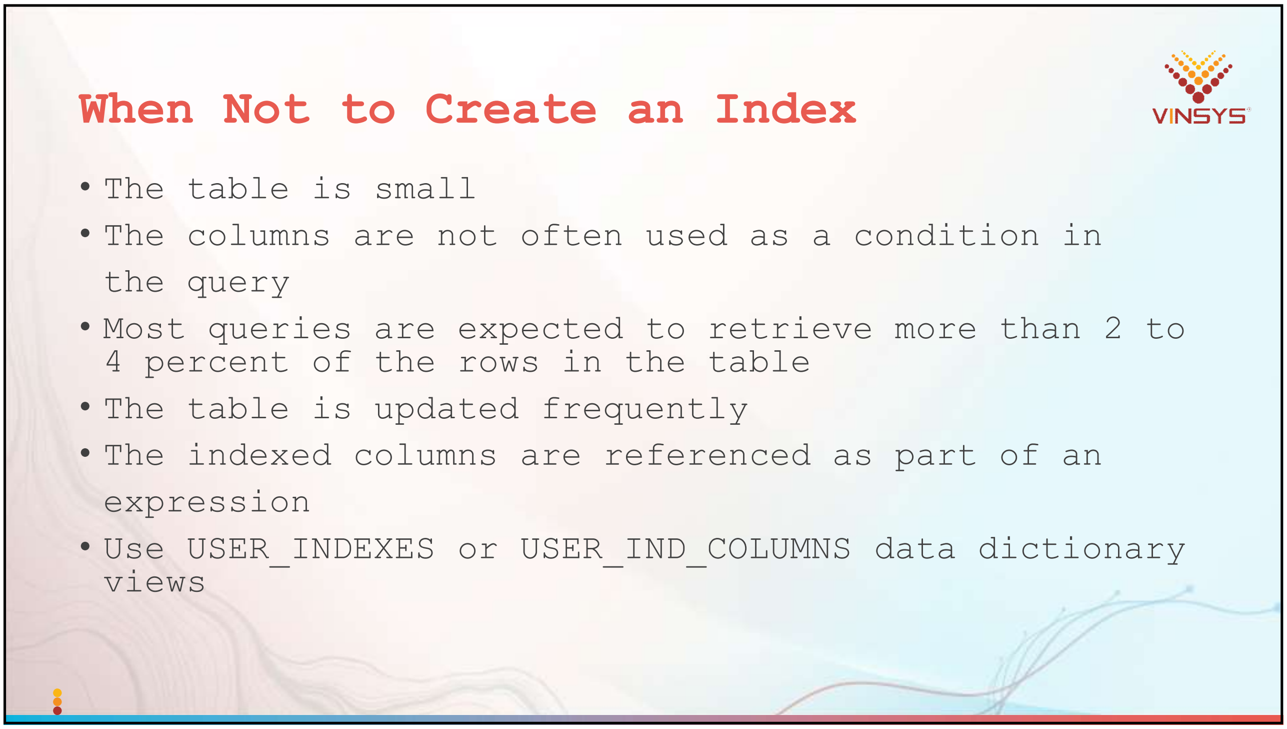
- Improve the speed of query access to the

```
LAST_NAME column in the EMPLOYEES table.
CREATE INDEX emp_last_name_idx
ON emp (ename);
Index created.
```



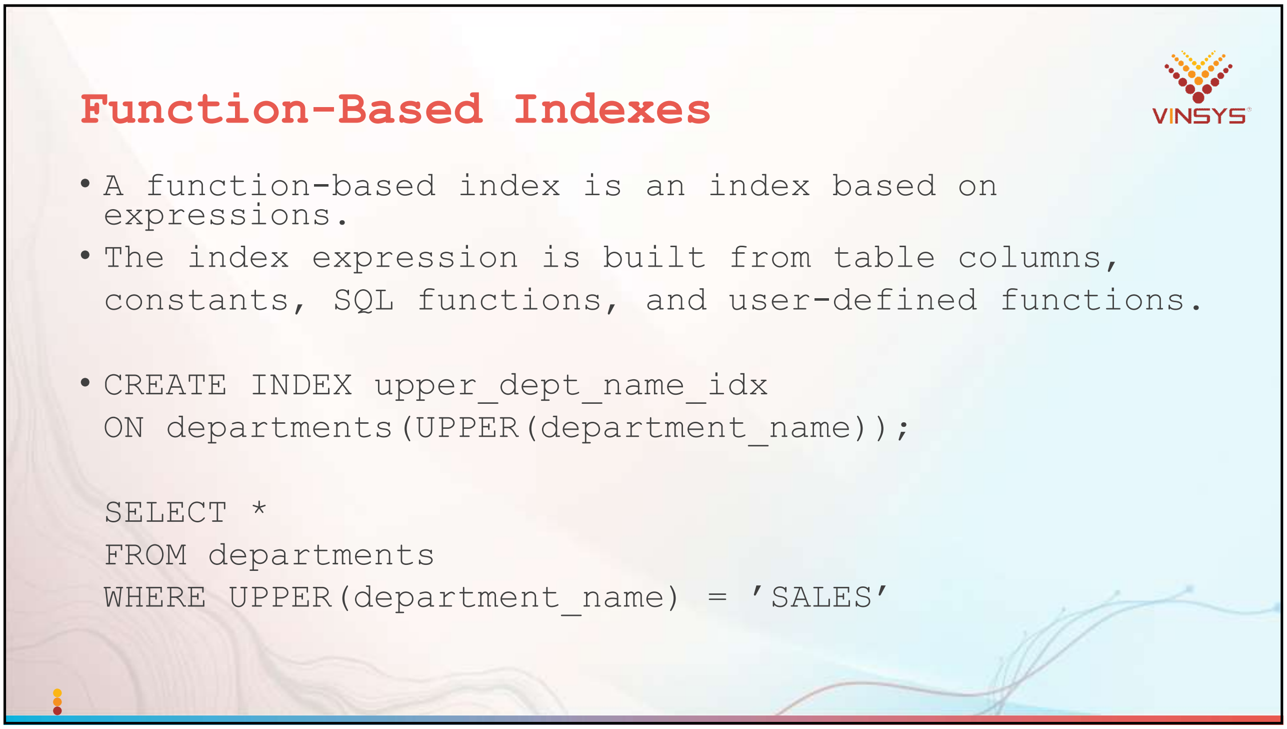
When to Create an Index

- You should create an index if:
- A column contains a wide range of values
- A column contains a large number of null values
- One or more columns are frequently used together in a WHERE clause or a join condition
- The table is large and most queries are expected to retrieve less than 2 to 4 percent of the rows



When Not to Create an Index

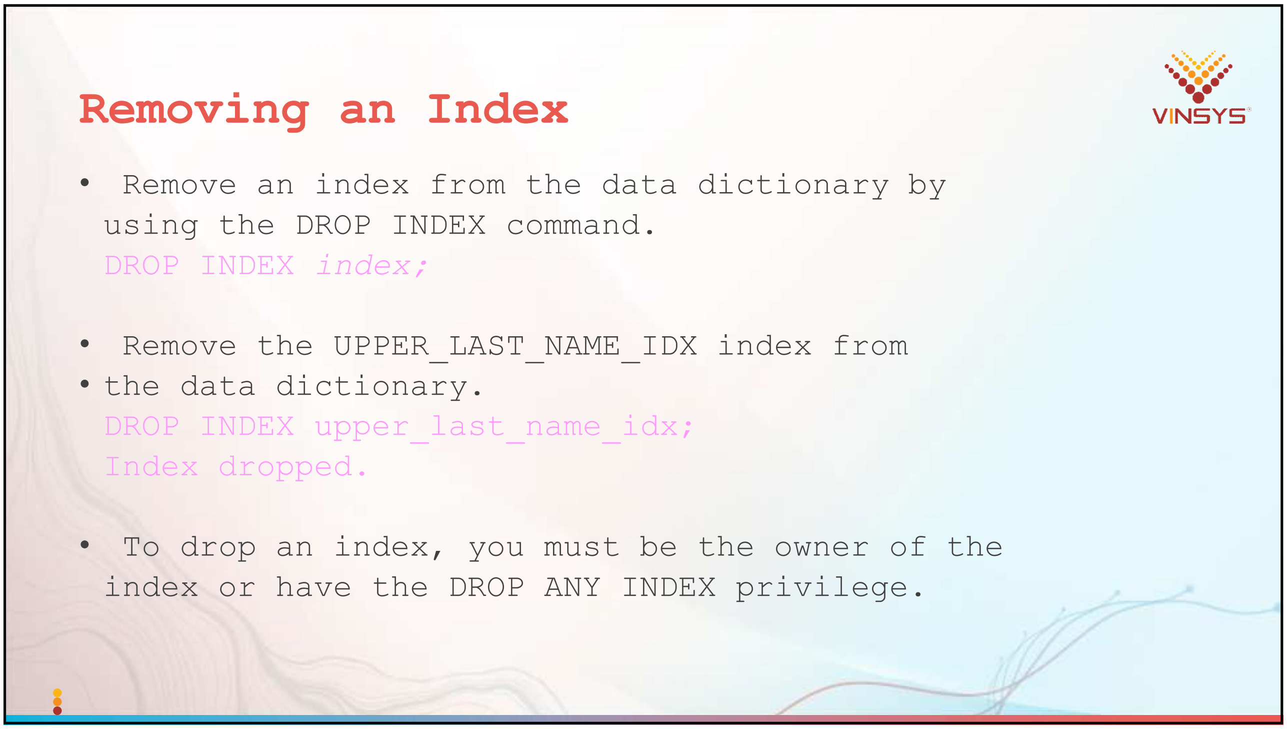
- The table is small
- The columns are not often used as a condition in the query
- Most queries are expected to retrieve more than 2 to 4 percent of the rows in the table
- The table is updated frequently
- The indexed columns are referenced as part of an expression
- Use USER_INDEXES or USER_IND_COLUMNS data dictionary views



Function-Based Indexes

- A function-based index is an index based on expressions.
- The index expression is built from table columns, constants, SQL functions, and user-defined functions.

-
  ```
  CREATE INDEX upper_dept_name_idx
  ON departments(UPPER(department_name));

  SELECT *
  FROM departments
  WHERE UPPER(department_name) = 'SALES'
  ```



Removing an Index

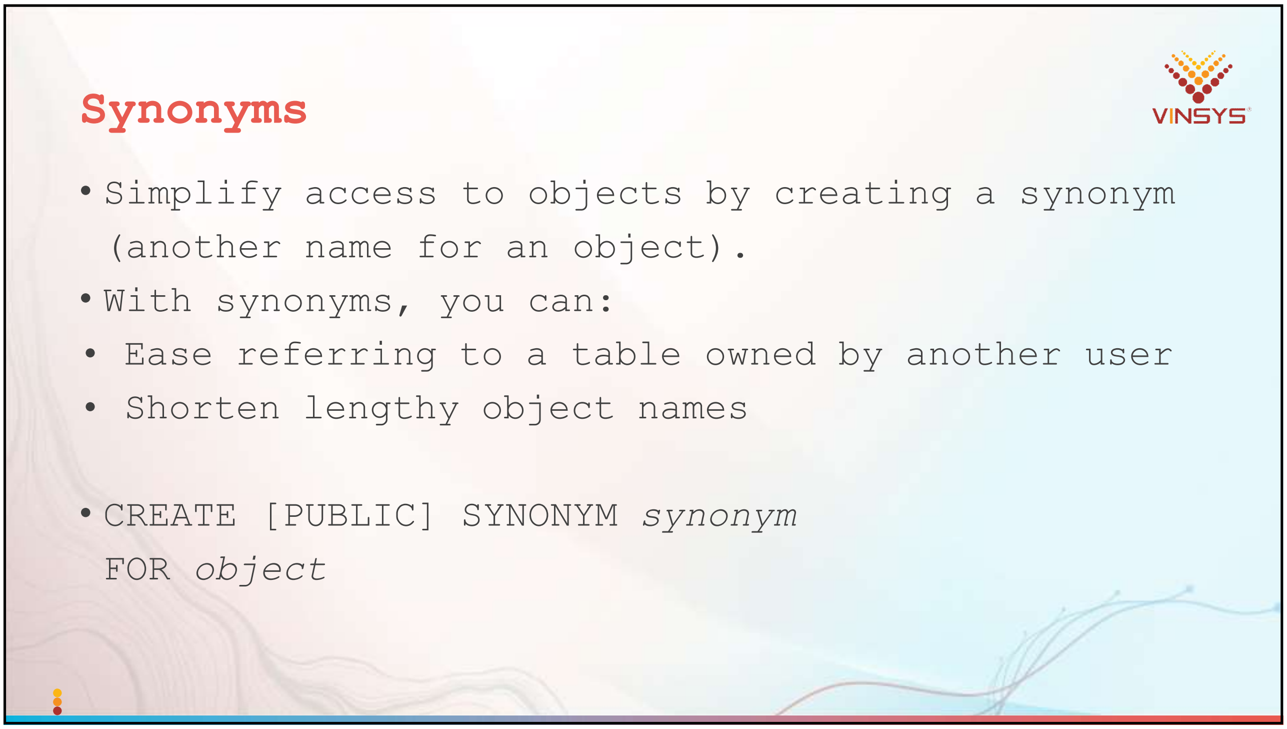
- Remove an index from the data dictionary by using the DROP INDEX command.

  ```
  DROP INDEX index;
  ```

- Remove the UPPER_LAST_NAME_IDX index from
- the data dictionary.

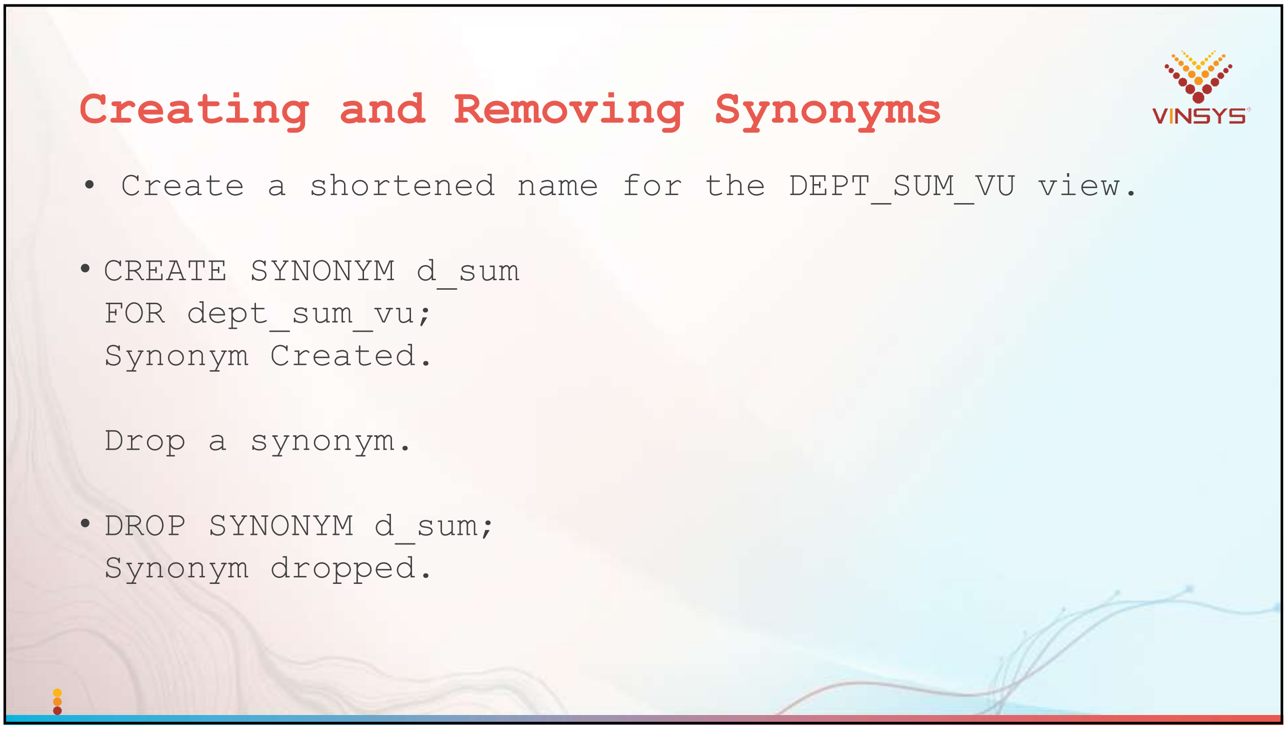
  ```
  DROP INDEX upper_last_name_idx;
  Index dropped.
  ```

- To drop an index, you must be the owner of the index or have the DROP ANY INDEX privilege.



Synonyms

- Simplify access to objects by creating a synonym (another name for an object).
- With synonyms, you can:
  - Ease referring to a table owned by another user
  - Shorten lengthy object names
- CREATE [PUBLIC] SYNONYM *synonym*
  FOR *object*



Creating and Removing Synonyms

- Create a shortened name for the DEPT_SUM_VU view.

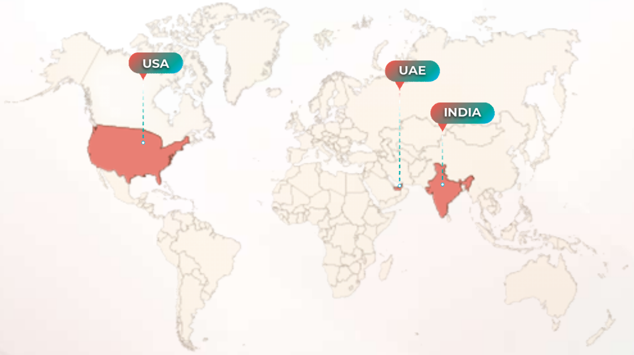
```
CREATE SYNONYM d_sum
FOR dept_sum_vu;
Synonym Created.
```

Drop a synonym.

```
DROP SYNONYM d_sum;
Synonym dropped.
```

Physical Presence

VINSYS



www.vinsys.com

enquiry@vinsys.com

Shivaji Niketan, Tejas Society,
Behind Kothrud Bus Stand, Near
Mantri Park, Kothrud, Pune -
411029.

304, City Tower 2, Near Crown
Plaza, Sheikh Zayed Road. Dubai,
UAE.
PO.Box - 213279

132 West 31st Street, First
Floor, New York, 10001, USA