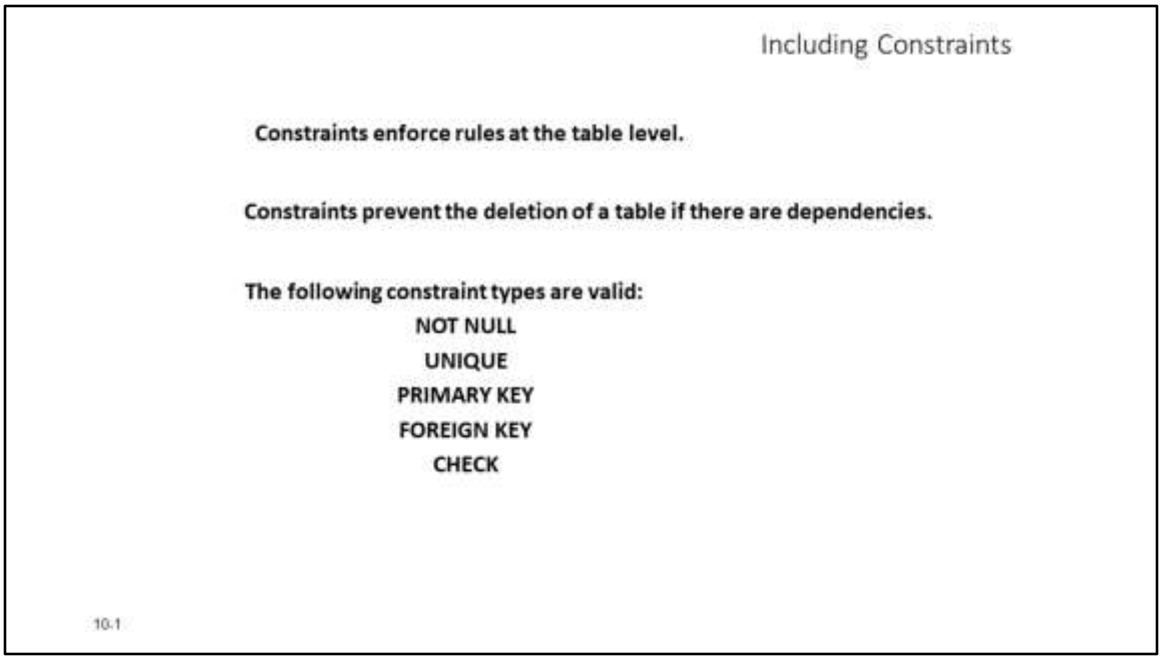


Including Constraints

Constraints enforce rules at the table level.

Constraints prevent the deletion of a table if there are dependencies.

The following constraint types are valid:

NOT NULL
UNIQUE
PRIMARY KEY
FOREIGN KEY
CHECK

10-1

Constraints

The Oracle server uses constraints to prevent invalid data entry into tables.

You can use constraints to do the following:

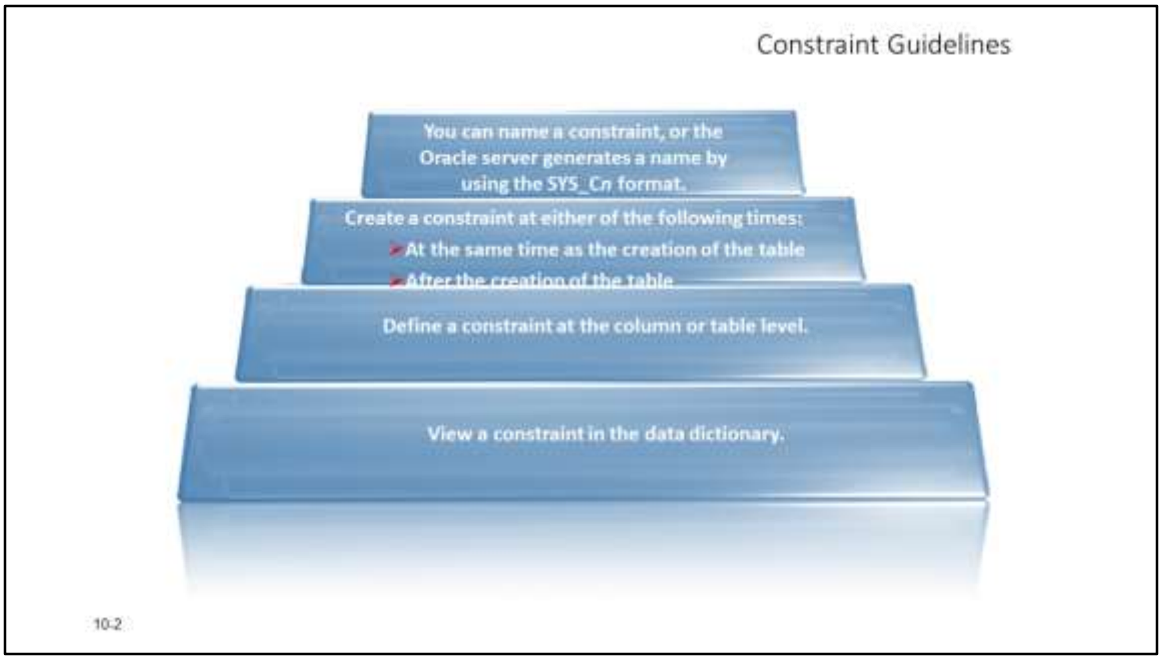
Enforce rules on the data in a table whenever a row is inserted, updated, or deleted from that table. The constraint must be satisfied for the operation to succeed.

Prevent the deletion of a table if there are dependencies from other tables.

Provide rules for Oracle tools, such as Oracle Developer.

Data Integrity Constraints

Constraint	**Description**
`NOT NULL`	Specifies that the column cannot contain a null value
`UNIQUE`	Specifies a column or combination of columns whose values must be unique for all rows in the table
`PRIMARY KEY`	Uniquely identifies each row of the table
`FOREIGN KEY`	Establishes and enforces a referential integrity between the column and a column of the referenced table such that values in one table match values in another table.
`CHECK`	Specifies a condition that must be true


Constraint Guidelines

You can name a constraint, or the Oracle server generates a name by using the SYS_Cn format.

Create a constraint at either of the following times:
- At the same time as the creation of the table
- After the creation of the table

Define a constraint at the column or table level.

View a constraint in the data dictionary.

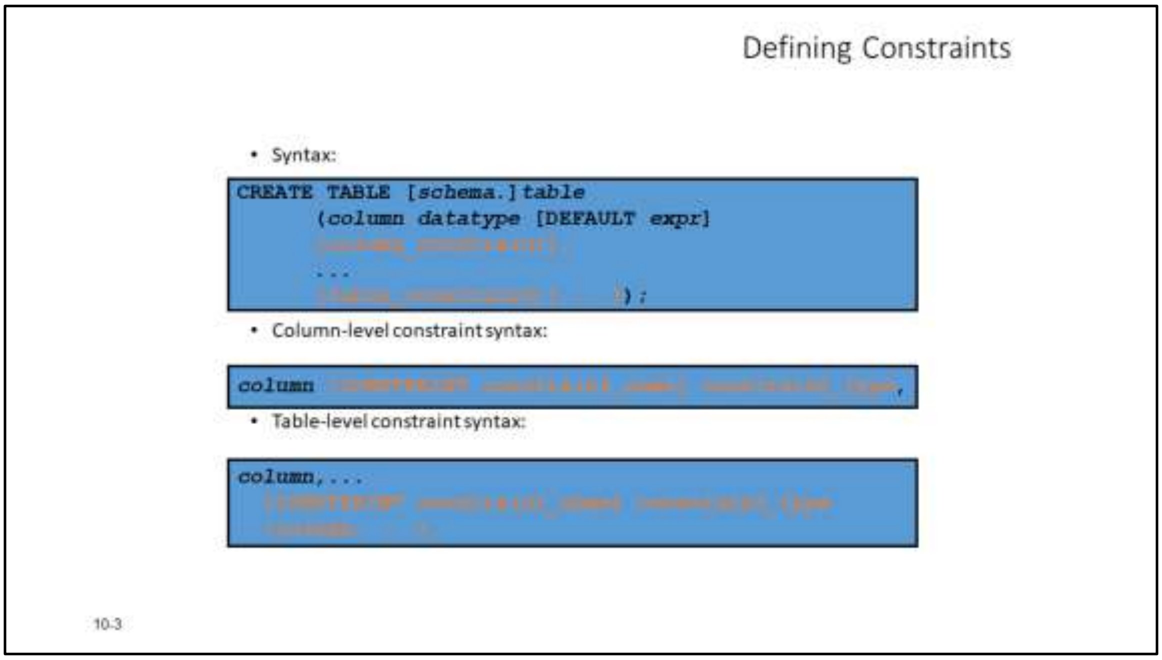
10-2

Constraint Guidelines

All constraints are stored in the data dictionary. Constraints are easy to reference if you give them a meaningful name. Constraint names must follow the standard object-naming rules, except that the name cannot be the same as another object owned by the same user. If you do not name your constraint, the Oracle server generates a name with the format `SYS_Cn`, where *n* is an integer so that the constraint name is unique.

Constraints can be defined at the time of table creation or after the creation of the table. You can define a constraint at the column or table level. Functionally, a table-level constraint is the same as a column-level constraint.

For more information, see the section on "Constraints" in *Oracle Database SQL Language Reference*
for 10*g* or 11*g* database.



Defining Constraints

- Syntax:

```
CREATE TABLE [schema.]table
      (column datatype [DEFAULT expr]
      [column_constraint],
      ...
      [table_constraint][,...]);
```

- Column-level constraint syntax:

```
column [CONSTRAINT constraint_name] constraint_type,
```

- Table-level constraint syntax:

```
column,...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

10-3

Defining Constraints

The slide gives the syntax for defining constraints when creating a table. You can create constraints at either the column level or table level. Constraints defined at the column level are included when the column is defined. Table-level constraints are defined at the end of the table definition and must refer to the column or columns on which the constraint pertains in a set of parentheses. It is mainly the syntax that differentiates the two; otherwise, functionally, a column-level constraint is the same as a table-level constraint.

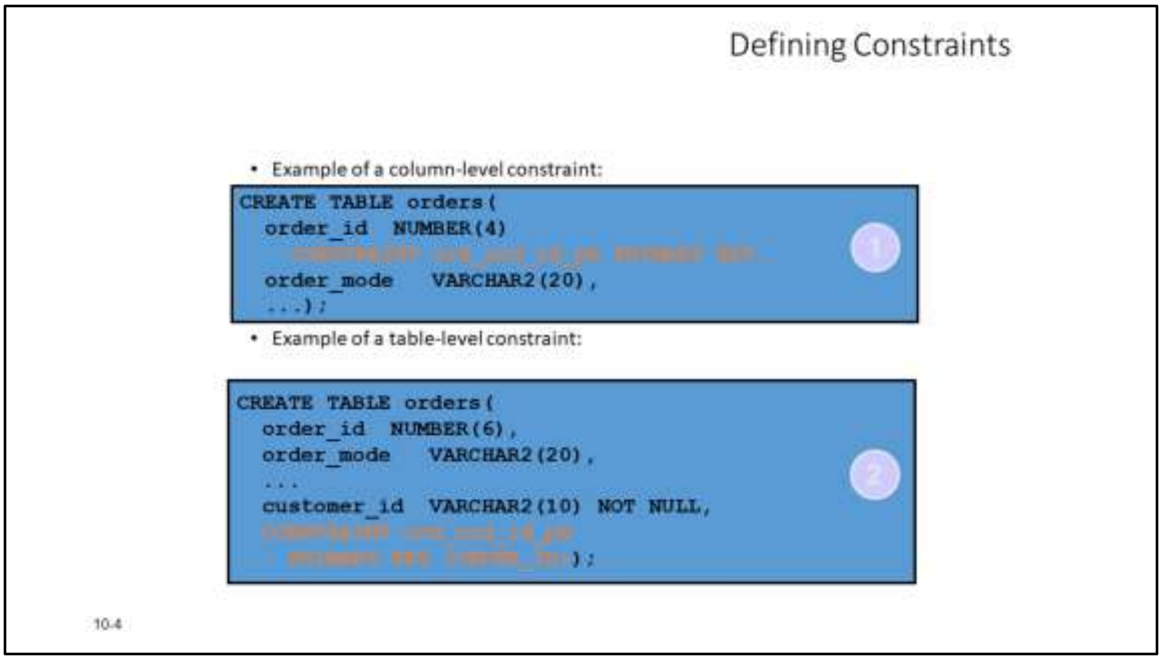
`NOT NULL` constraints must be defined at the column level.

Constraints that apply to more than one column must be defined at the table level.

In the syntax:

`schema`	Is the same as the owner's name
`table`	Is the name of the table
`DEFAULT expr`	Specifies a default value to be used if a value is omitted in the `INSERT` statement
`column`	Is the name of the column
`datatype`	Is the column's data type and length
`column_constraint`	Is an integrity constraint as part of the column definition

`table_constraint`	Is an integrity constraint as part of the table definition



Defining Constraints

- Example of a column-level constraint:

```
CREATE TABLE orders(
  order_id  NUMBER(4)
    [illegible]
  order_mode   VARCHAR2(20),
  ...);
```

- Example of a table-level constraint:

```
CREATE TABLE orders(
  order_id  NUMBER(6),
  order_mode   VARCHAR2(20),
  ...
  customer_id  VARCHAR2(10) NOT NULL,
  [illegible]
  [illegible]);
```

10-4

Defining Constraints (continued)

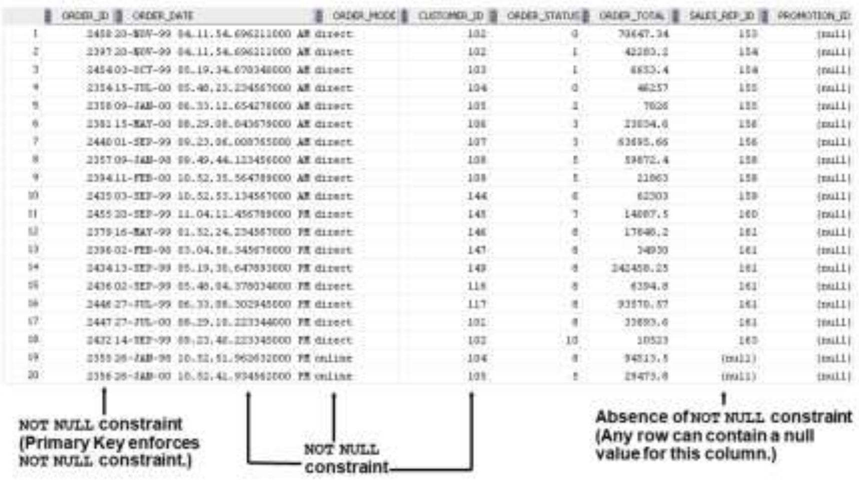
Constraints are usually created at the same time as the table. Constraints can be added to a table after its creation and also be temporarily disabled.

Both examples in the slide create a primary key constraint on the `EMPLOYEE_ID` column of the `EMPLOYEES` table.

1. The first example uses the column-level syntax to define the constraint.
2. The second example uses the table-level syntax to define the constraint.

More details about the primary key constraint are provided later in this lesson.

NOT NULL Constraint Orders table



	ORDER_ID	ORDER_DATE	ORDER_MODE	CUSTOMER_ID	ORDER_STATUS	ORDER_TOTAL	SALES_REP_ID	PROMOTION_ID
1	2458	20-NOV-99 04.11.54.696211000 AM	direct	102	0	70647.34	153	(null)
2	2397	20-NOV-99 04.11.54.696211000 AM	direct	102	1	42283.2	154	(null)
3	2454	03-OCT-99 08.19.34.670348000 AM	direct	103	1	6653.4	154	(null)
4	2354	15-JUL-00 05.40.23.234567000 AM	direct	104	0	46257	155	(null)
5	2358	09-JAN-00 06.33.12.654278000 AM	direct	105	2	7826	155	(null)
6	2381	15-MAY-00 08.29.08.043679000 AM	direct	106	3	23034.6	156	(null)
7	2440	01-SEP-99 09.23.06.008765000 AM	direct	107	3	63695.66	156	(null)
8	2357	09-JAN-98 09.49.44.123456000 AM	direct	108	5	59872.4	158	(null)
9	2394	11-FEB-00 10.52.35.564789000 AM	direct	108	5	21863	158	(null)
10	2435	03-SEP-99 10.52.53.134567000 AM	direct	144	6	62303	159	(null)
11	2455	20-SEP-99 11.04.11.456789000 PM	direct	145	7	14087.5	160	(null)
12	2379	16-MAY-99 01.32.24.234567000 PM	direct	146	8	17848.2	161	(null)
13	2396	02-FEB-98 03.04.58.345678000 PM	direct	147	8	34930	161	(null)
14	2434	13-SEP-99 05.19.38.647893000 PM	direct	149	8	242450.25	161	(null)
15	2436	02-SEP-99 05.48.04.378034000 PM	direct	116	8	6394.8	161	(null)
16	2446	27-JUL-99 06.33.08.302945000 PM	direct	117	8	93570.57	161	(null)
17	2447	27-JUL-00 08.29.18.223344000 PM	direct	101	8	33893.6	161	(null)
18	2432	14-SEP-99 09.23.48.223345000 PM	direct	102	10	10523	163	(null)
19	2355	26-JAN-98 10.32.51.962632000 PM	online	104	8	94513.5	(null)	(null)
20	2356	26-JAN-00 10.52.41.934562000 PM	online	105	5	29473.8	(null)	(null)

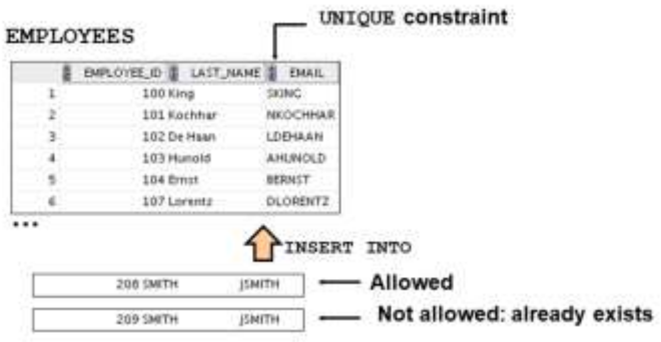
10-5

NOT NULL Constraint

The NOT NULL constraint ensures that the column contains no null values. Columns without the NOT NULL constraint can contain null values by default. NOT NULL constraints must be defined at the column level. In theORDERS table, the ORDER_ID column inherits a NOT NULL constraint as it is defined as a primary key. Otherwise, the ORDER_DATE, ORDER_NAME, CUSTOMER_ID columns have the NOT NULL constraint enforced on them.

Note: Primary key constraint is discussed in detail later in this lesson.

UNIQUE Constraint



	EMPLOYEE_ID	LAST_NAME	EMAIL
1	100	King	SKING
2	101	Kochhar	NKOCHHAR
3	102	De Haan	LDEHAAN
4	103	Hunold	AHUNOLD
5	104	Ernst	BERNST
6	107	Lorentz	DLORENTZ

...

208	SMITH	JSMITH
209	SMITH	JSMITH

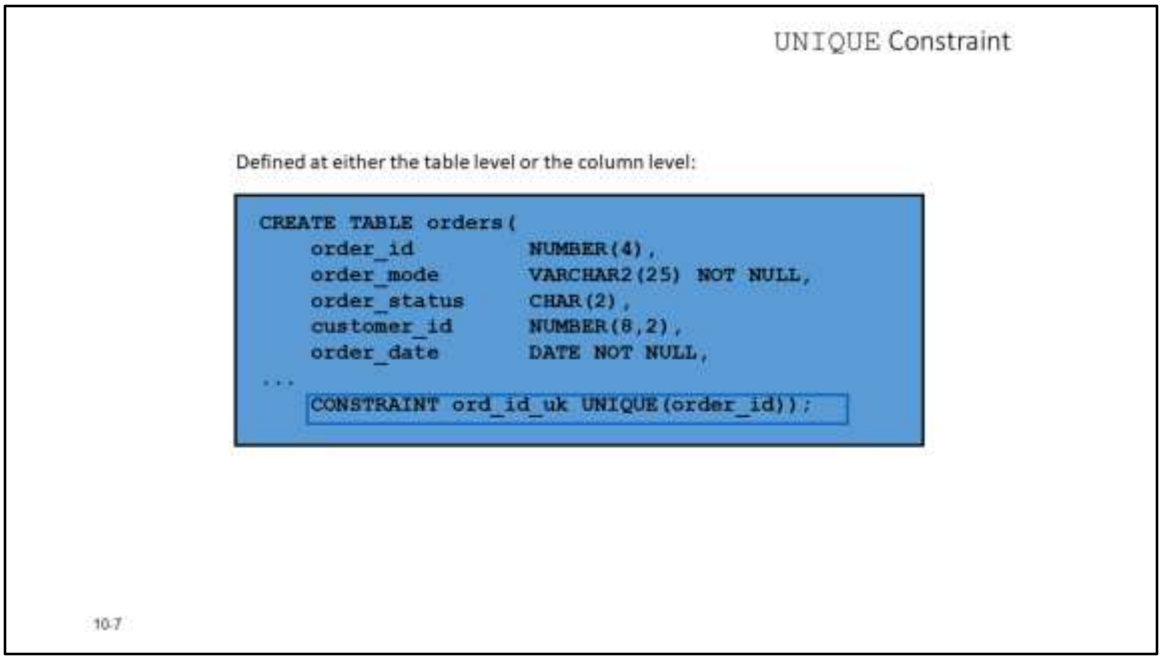
10-6

UNIQUE Constraint

A UNIQUE key integrity constraint requires that every value in a column or a set of columns (key) be unique—that is, no two rows of a table can have duplicate values in a specified column or a set of columns. The column (or set of columns) included in the definition of the UNIQUE key constraint is called the *unique key*. If the UNIQUE constraint comprises more than one column, that group of columns is called a *composite unique key*.

UNIQUE constraints enable the input of nulls unless you also define NOT NULL constraints for the same columns. In fact, any number of rows can include nulls for columns without the NOT NULL constraints because nulls are not considered equal to anything. A null in a column (or in all columns of a composite UNIQUE key) always satisfies a UNIQUE constraint.

Note: Because of the search mechanism for the UNIQUE constraints on more than one column, you cannot have identical values in the non-null columns of a partially null composite UNIQUE key constraint.


UNIQUE Constraint

Defined at either the table level or the column level:

```
CREATE TABLE orders(
    order_id        NUMBER(4),
    order_mode      VARCHAR2(25) NOT NULL,
    order_status    CHAR(2),
    customer_id     NUMBER(8,2),
    order_date      DATE NOT NULL,
...
    CONSTRAINT ord_id_uk UNIQUE(order_id));
```

10-7

UNIQUE Constraint (continued)

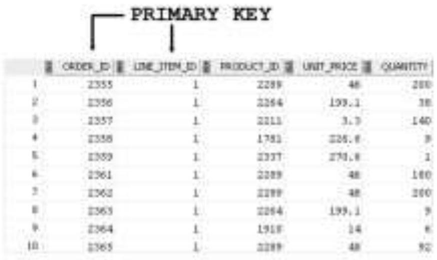
UNIQUE constraints can be defined at the column level or table level. You define the constraint at the table level when you want to create a composite unique key. A composite key is defined when there is not a single attribute that can uniquely identify a row. In that case, you can have a unique key that is composed of two or more columns, the combined value of which is always unique and can identify rows.

The example in the slide applies the UNIQUE constraint to the ORDER_ID column of the ORDERS table. The name of the constraint is ORD_ID_UK.

Note: The Oracle server enforces the UNIQUE constraint by implicitly creating a unique index on the unique key column or columns.

PRIMARY KEY Constraint

ORDER_ITEMS



	ORDER_ID	LINE_ITEM_ID	PRODUCT_ID	UNIT_PRICE	QUANTITY
1	2355	1	2289	46	200
2	2356	1	2264	199.1	38
3	2357	1	2211	3.3	140
4	2358	1	1781	226.6	9
5	2359	1	2337	270.6	1
6	2361	1	2289	46	180
7	2362	1	2289	48	200
8	2363	1	2264	199.1	9
9	2364	1	1910	14	6
10	2365	1	2289	48	92

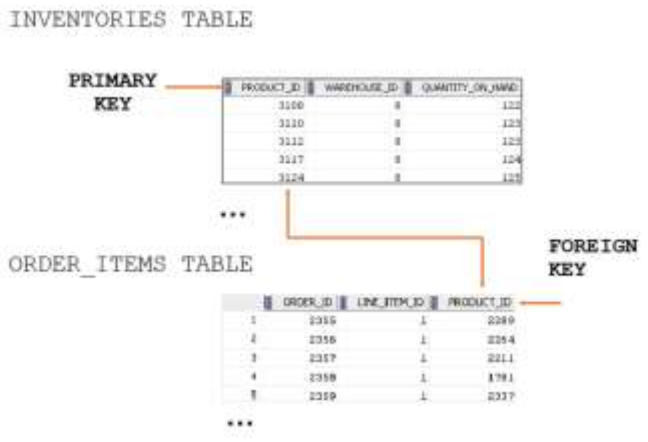
10-8

`PRIMARY KEY` Constraint

A `PRIMARY KEY` constraint creates a primary key for the table. Only one primary key can be created for each table. The `PRIMARY KEY` constraint is a column or a set of columns that uniquely identifies each row in a table. This constraint enforces the uniqueness of the column or column combination and ensures that no column that is part of the primary key can contain a null value.

Note: Because uniqueness is part of the primary key constraint definition, the Oracle server enforces the uniqueness by implicitly creating a unique index on the primary key column or columns.

FOREIGN KEY Constraint



PRODUCT_ID	WAREHOUSE_ID	QUANTITY_ON_HAND
3100	8	122
3110	8	123
3112	8	123
3117	8	124
3124	8	125

	ORDER_ID	LINE_ITEM_ID	PRODUCT_ID
1	2355	1	2289
2	2356	1	2264
3	2357	1	2211
4	2358	1	1781
5	2359	1	2337

10-9

FOREIGN KEY Constraint

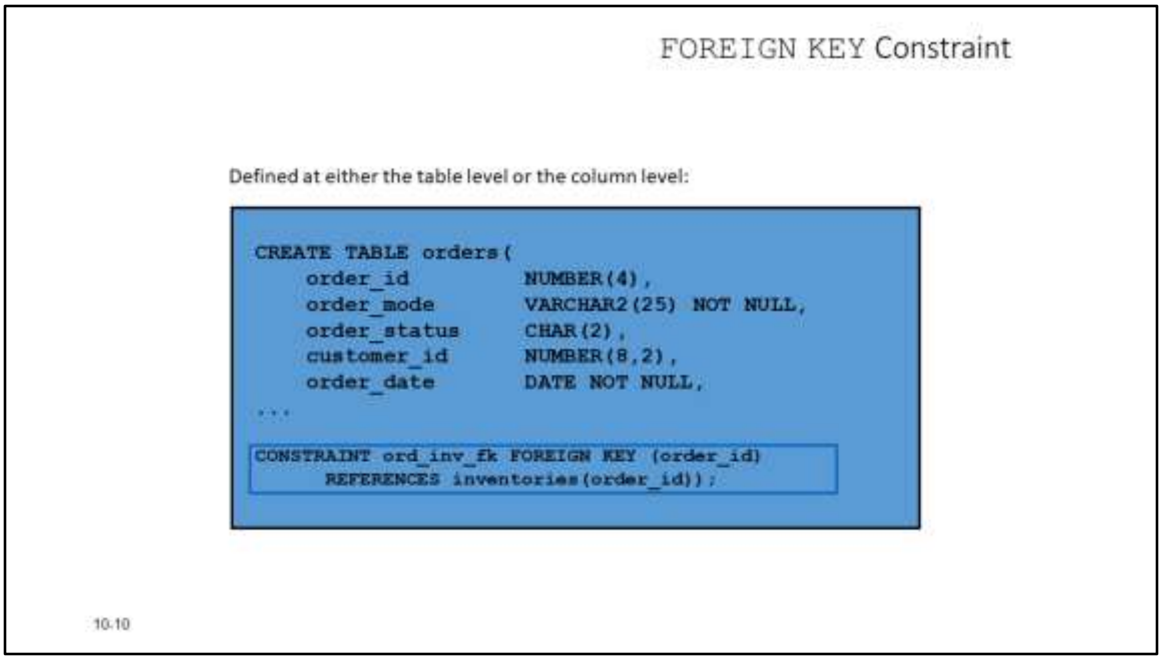
The FOREIGN KEY (or referential integrity) constraint designates a column or a combination of columns as a foreign key and establishes a relationship with a primary key or a unique key in the same table or a different table.

In the example in the slide, product_id has been defined as the foreign key in the ORDER_ITEMS table (dependent or child table); it references the product_ID column of the ORDER_ITEMS table (the referenced or parent table).

Guidelines

A foreign key value must match an existing value in the parent table or be NULL.

Foreign keys are based on data values and are purely logical, rather than physical, pointers.



FOREIGN KEY Constraint

Defined at either the table level or the column level:

```
CREATE TABLE orders(
    order_id         NUMBER(4),
    order_mode       VARCHAR2(25) NOT NULL,
    order_status     CHAR(2),
    customer_id      NUMBER(8,2),
    order_date       DATE NOT NULL,
...

CONSTRAINT ord_inv_fk FOREIGN KEY (order_id)
      REFERENCES inventories(order_id));
```

10-10

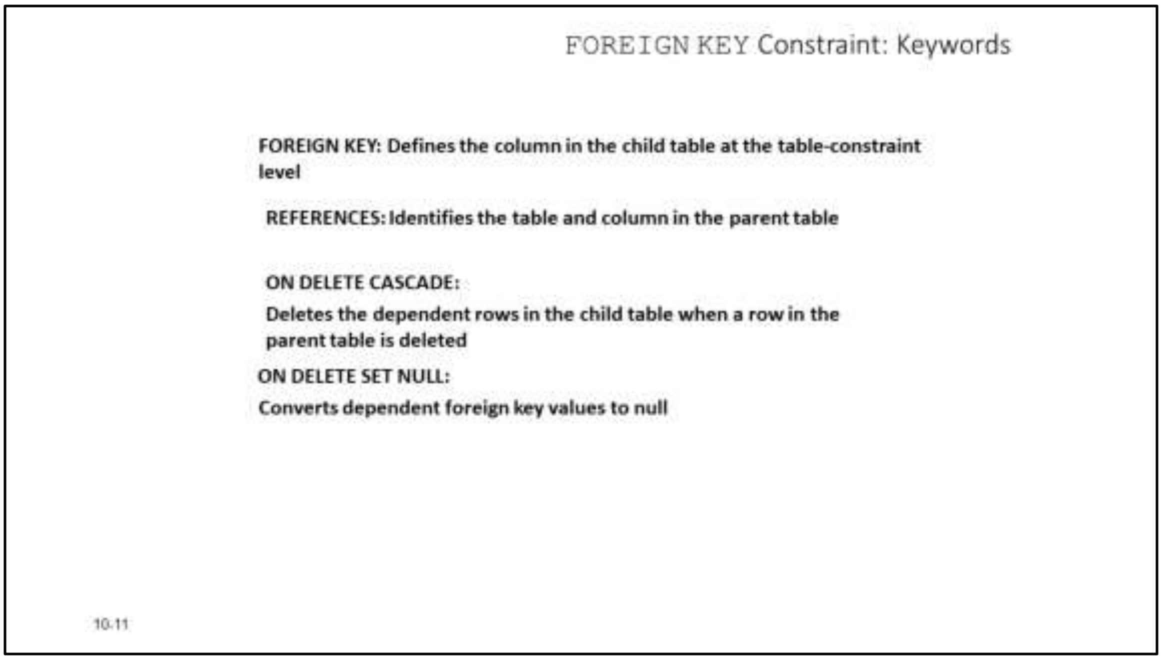
FOREIGN KEY Constraint (continued)

FOREIGN KEY constraints can be defined at the column or table constraint level. A composite foreign key must be created by using the table-level definition.

The example in the slide defines a FOREIGN KEY constraint on the ORDER_ID column of the ORDER_ITEMS table, using table-level syntax. The name of the constraint is ORD_INV_FK.

The foreign key can also be defined at the column level, provided that the constraint is based on a single column. The syntax differs in that the keywords FOREIGN KEY do not appear. For example:

```
CREATE TABLE orders
(...
order_id NUMBER(4) CONSTRAINT ord_inv_fk
REFERENCES inventories(order_id)
...
)
```



FOREIGN KEY Constraint: Keywords

FOREIGN KEY: Defines the column in the child table at the table-constraint level

REFERENCES: Identifies the table and column in the parent table

ON DELETE CASCADE:
Deletes the dependent rows in the child table when a row in the parent table is deleted

ON DELETE SET NULL:
Converts dependent foreign key values to null

10-11

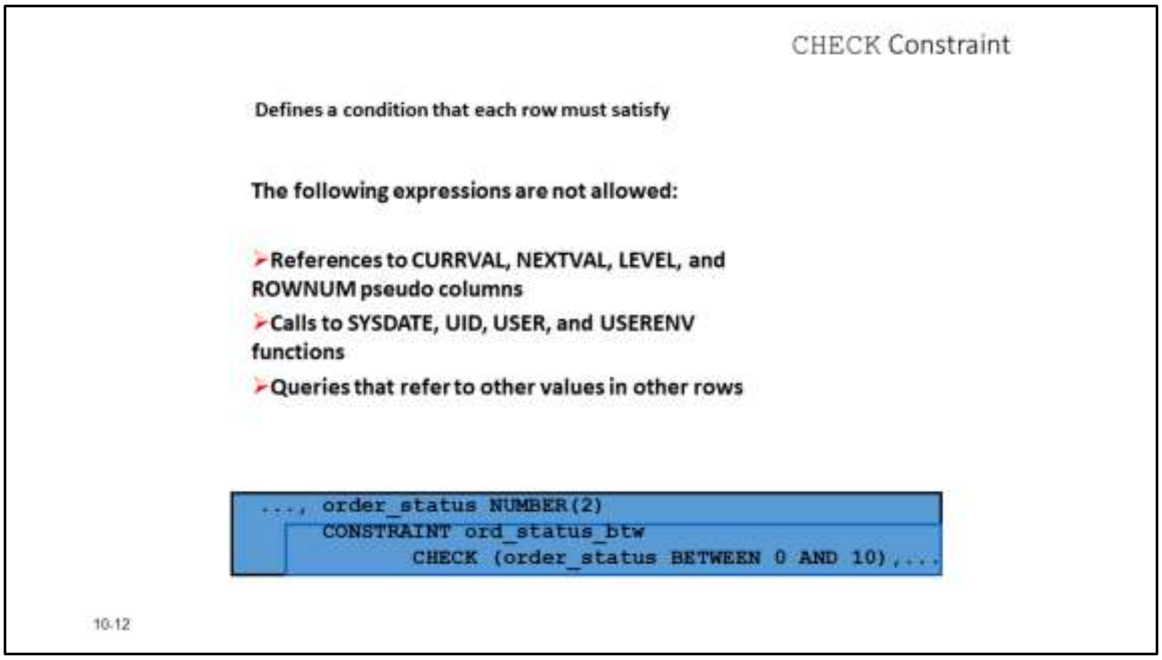
`FOREIGN KEY` Constraint: Keywords

The foreign key is defined in the child table and the table containing the referenced column is the parent table. The foreign key is defined using a combination of the following keywords:

- `FOREIGN KEY` is used to define the column in the child table at the table-constraint level.
- `REFERENCES` identifies the table and the column in the parent table.
- `ON DELETE CASCADE` indicates that when a row in the parent table is deleted, the dependent rows in the child table are also deleted.
- `ON DELETE SET NULL` indicates that when a row in the parent table is deleted, the foreign key values are set to null.

The default behavior is called the *restrict rule*, which disallows the update or deletion of referenced data.

Without the `ON DELETE CASCADE` or the `ON DELETE SET NULL` options, the row in the parent table cannot be deleted if it is referenced in the child table.


CHECK Constraint

Defines a condition that each row must satisfy

The following expressions are not allowed:

- References to CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudo columns
- Calls to SYSDATE, UID, USER, and USERENV functions
- Queries that refer to other values in other rows

```
..., order_status NUMBER(2)
     CONSTRAINT ord_status_btw
          CHECK (order_status BETWEEN 0 AND 10),...
```

10-12

CHECK Constraint

The CHECK constraint defines a condition that each row must satisfy. The condition can use the same constructs as the query conditions, with the following exceptions:

References to the CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudocolumns

Calls to SYSDATE, UID, USER, and USERENV functions

Queries that refer to other values in other rows

A single column can have multiple CHECK constraints that refer to the column in its definition. There is no limit to the number of CHECK constraints that you can define on a column.

CHECK constraints can be defined at the column level or table level.

```
CREATE TABLE orders
  (...
   order_status NUMBER(8,2) CONSTRAINT ord_status_btw
                 CHECK (order_status BETWEEN 0 AND 10)
  ...
  )
```