



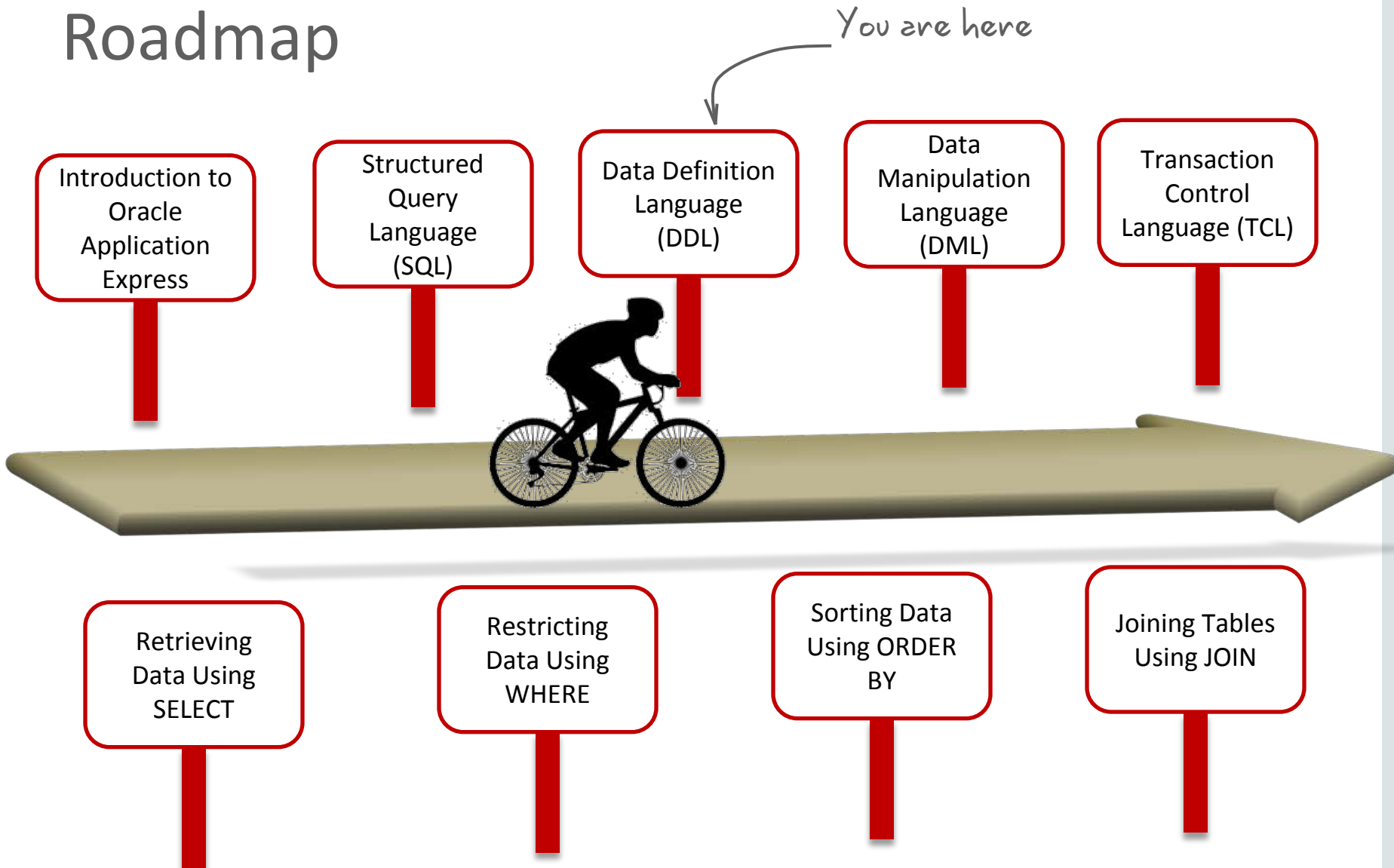
Database Foundations

6-3

Data Definition Language (DDL)



Roadmap



Objectives

This lesson covers the following objectives:

- Identify the steps needed to create database tables
- Describe the purpose of the data definition language (DDL)
- List the DDL operations needed to build and maintain a database's tables



Database Objects

Object	Description
Table	Is the basic unit of storage; consists of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives an alternative name to an object

Naming Rules for Tables and Columns

Table names and column names must:

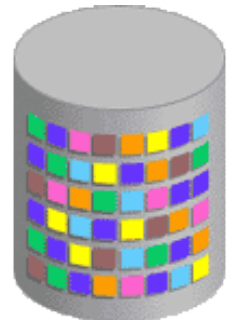
- Begin with a letter
- Be 1–30 characters long
- Contain only A–Z, a–z, 0–9, _, \$, and #
- Not duplicate the name of another object owned by the same user
- Not be an Oracle server–reserved word

CREATE TABLE Statement

- To issue a `CREATE TABLE` statement, you must have:
 - The `CREATE TABLE` privilege
 - A storage area

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

- Specify in the statement:
 - Table name
 - Column name, column data type, column size
 - Integrity constraints (optional)
 - Default values (optional)



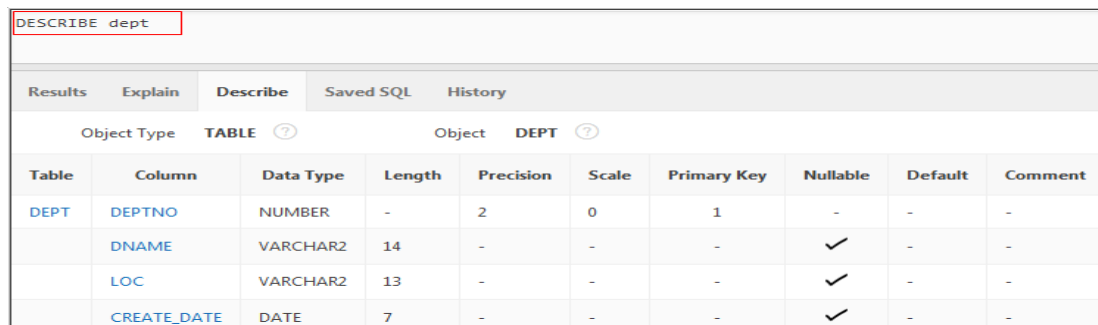
Creating Tables

- Create the table:

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

- Confirm table creation:

```
DESCRIBE dept
```



The screenshot shows the Oracle SQL Developer interface with the command 'DESCRIBE dept' entered in the command window. Below the command window, the 'Describe' tab is active, displaying the table structure for 'DEPT'.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT	DEPTNO	NUMBER	-	2	0	1	-	-	-
	DNAME	VARCHAR2	14	-	-	-	✓	-	-
	LOC	VARCHAR2	13	-	-	-	✓	-	-
	CREATE_DATE	DATE	7	-	-	-	✓	-	-

Data Types

Data Type	Description
VARCHAR2(size)	Variable-length character data (A maximum size must be specified; minimum size is 1.) Maximum size: 32767 bytes if MAX_SQL_STRING_SIZE = EXTENDED 4000 bytes if MAX_SQL_STRING_SIZE = LEGACY
CHAR(size)	Fixed-length character data of length (size) bytes. (Default and minimum size is 1; maximum size is 2,000)
NUMBER(p, s)	Variable-length numeric data. Precision is p, and scale is s. (Precision is the total number of decimal digits, and scale is the number of digits to the right of the decimal point; precision can range from 1 to 38, and scale can range from -84 to 127.)
DATE	Date and time values to the nearest second between January 1, 4712 B.C, and December 31, 9999 A.D.
LONG	Variable-length character data (up to 2 GB)

Data Types

Data Type	Description
CLOB	A character large object (CLOB) containing single-byte or multibyte characters. Maximum size is $(4 \text{ GB} - 1) * (\text{DB_BLOCK_SIZE})$; stores national character set data.
NCLOB	A CLOB containing Unicode characters. Both fixed-width and variable-width character sets are supported, both using the database national character set. Maximum size is $(4 \text{ GB} - 1) * (\text{database block size})$; stores national character set data.
RAW (Size)	Raw binary data of length <i>size</i> bytes. You must specify <i>size</i> for a RAW value. Maximum <i>size</i> : 32767 bytes if MAX_SQL_STRING_SIZE = EXTENDED 4000 bytes if MAX_SQL_STRING_SIZE = LEGACY
LONG RAW	Raw binary data of variable length up to 2 GB.
BLOB	A binary large object. Maximum size is $(4 \text{ GB} - 1) * (\text{DB_BLOCK_SIZE initialization parameter (8 TB to 128 TB)})$.
BFILE	Binary data stored in an external file (up to 4 GB).
ROWID	Base 64 string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudocolumn

Example: Creating a Table with Different Data Types

```
CREATE TABLE print_media
(product_id NUMBER(6),
 media_id NUMBER(6),
 media_desc VARCHAR2(100),
 media_composite BLOB,
 media_sourcetext CLOB,
 media_finaltext CLOB,
 media_photo BLOB,
 media_graphic BFILE);
```

Date Data Types

Data Type	Description
TIMESTAMP	Enables storage of time as a date with fractional seconds. It stores the year, month, day, hour, minute, the second value of the DATE data types, and the fractional seconds value. There are several variations of this data type, such as WITH TIMEZONE and WITH LOCALTIMEZONE.
INTERVAL YEAR TO MONTH	Enables storage of time as an interval of years and months. Used to represent the difference between two datetime values in which the only significant portions are the year and month.
INTERVAL DAY TO SECOND	Enables storage of time as an interval of days, hours, minutes, and seconds; used to represent the precise difference between two datetime values.
TIMESTAMP WITH TIME ZONE	Variant of TIMESTAMP that includes a time zone region name or time zone offset in its value.



Examples: Date Data Types

- Example of TIMESTAMP data type:

```
CREATE TABLE table_ts(c_id NUMBER, c_ts TIMESTAMP);  
INSERT INTO table_ts VALUES(1, '01-JAN-2003 2:00:00');
```

- Example of a table with TIMESTAMP, INTERVAL YEAR TO MONTH and INTERVAL DAY TO SECOND columns:

```
CREATE TABLE time_table  
(start_time      TIMESTAMP,  
 duration_1      INTERVAL DAY (6) TO SECOND (5),  
 duration_2      INTERVAL YEAR TO MONTH);
```

DEFAULT Option

- Specify a default value for a column during CREATE TABLE.

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- Literal values, expressions, or SQL functions are legal values.
- Another column's name or a pseudocolumn are illegal values.
- The default data type must match the column data type.

```
CREATE TABLE hire_dates  
  (id          NUMBER(8),  
   hire_date DATE DEFAULT SYSDATE);  
table HIRE_DATES created.
```

Case Scenario: Creating Tables

How about creating the tables for the simplified library database?



Case Scenario: Creating Tables

```
CREATE TABLE BOOK_TRANSACTION
(
  TRANSACTION_ID VARCHAR2(6),
  TRANSACTION_DATE DATE DEFAULT SYSDATE,
  TRANSACTION_TYPE VARCHAR2(10),
  BOOK_ID VARCHAR2(6),
  MEMBER_ID VARCHAR2(6)
);
```

```
CREATE TABLE AUTHOR
(
  AUTHOR_ID VARCHAR2(6),
  AUTHOR_NAME VARCHAR2(60)
);
```

```
CREATE TABLE BOOKS
```

```
(
  BOOK_ID VARCHAR2(6),
  TITLE VARCHAR2(255),
  PUBLISHER_ID VARCHAR2(6),
  AUTHOR_ID VARCHAR2(6)
);
```

```
CREATE TABLE MEMBERS
```

```
(
  MEMBER_ID VARCHAR2(6),
  FIRST_NAME VARCHAR2(50),
  LAST_NAME VARCHAR2(50),
  STREET_ADDRESS VARCHAR2(50),
  CITY VARCHAR2(20),
  STATE VARCHAR2(2),
  ZIP VARCHAR2(10)
);
```

```
CREATE TABLE PUBLISHER
```

```
(
  PUBLISHER_ID VARCHAR2(6),
  PUBLISHER_NAME VARCHAR2(100)
);
```



Case Scenario: Creating Tables



Creating Tables

```
Rows 10 [dropdown] [?] [Clear Command] [Find Tab]

CREATE TABLE AUTHOR
(
  AUTHOR_ID VARCHAR2(6),
  AUTHOR_NAME VARCHAR2(60)
);

CREATE TABLE MEMBERS
(
  MEMBER_ID VARCHAR2(6),
  FIRST_NAME VARCHAR2(50),
  LAST_NAME VARCHAR2(50),
  STREET_ADDRESS VARCHAR2(50),
  CITY VARCHAR2(20),
  STATE VARCHAR2(2),
  ZIP VARCHAR2(10)
);

CREATE TABLE PUBLISHER
(
  PUBLISHER_ID VARCHAR2(6),
  PUBLISHER_NAME VARCHAR2(100) NOT NULL
);

CREATE TABLE BOOKS
(
  BOOK_ID VARCHAR2(6),
  TITLE VARCHAR2(255) NOT NULL,
  PUBLISHER_ID VARCHAR2(6),
  AUTHOR_ID VARCHAR2(6)
);

CREATE TABLE BOOK_TRANSACTIONS
```

Successful creation of tables

Results	Explain	Describe	Saved SQL	History
Table created.				
0.03 seconds				

Including Constraints

- Constraints enforce rules at the table level.
- Constraints ensure the consistency and integrity of the database.
- The following constraint types are valid:
 - NOT NULL
 - UNIQUE
 - PRIMARY KEY
 - FOREIGN KEY
 - CHECK



Data Integrity Constraints

Constraints	Description
NOT NULL	The column cannot contain a null value.
UNIQUE	The values for a column or a combination of columns must be unique for all rows in the table.
PRIMARY KEY	The column (or a combination of columns) must contain the unique AND IS NOT NULL value for all rows.
FOREIGN KEY	The column (or a combination of columns) must establish and enforce a reference to a column or a combination of columns in another (or the same) table.
CHECK	A condition must be true.

Constraint Guidelines

- Name a constraint (otherwise, the Oracle server generates a name in the SYS_Cn format).

Constraint	Type
SYS_C0014370	Primary Key

- 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.

Defining Constraints

- CREATE TABLE with 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, ...),
```

Examples: Defining Constraints

- Column-level constraint:

```
CREATE TABLE employees(  
  employee_id NUMBER(6)  
    CONSTRAINT emp_emp_id_pk PRIMARY KEY,  
  first_name VARCHAR2(20),  
  ...);
```

1

- Table-level constraint:

```
CREATE TABLE employees(  
  employee_id NUMBER(6),  
  first_name VARCHAR2(20),  
  ...  
  job_id VARCHAR2(10) NOT NULL,  
  CONSTRAINT emp_emp_id_pk  
    PRIMARY KEY (EMPLOYEE_ID));
```

2

NOT NULL Constraint

Ensures that null values are not permitted for the column:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	24000	-	90	SKING	515.123.4567	17-JUN-03
101	Neena	Kochhar	17000	-	90	NKOCHHAR	515.123.4568	21-SEP-05
102	Lex	De Haan	17000	-	90	LDEHAAN	515.123.4569	13-JAN-01
103	Alexander	Hunold	9000	-	60	AHUNOLD	590.423.4567	03-JAN-06
104	Bruce	Ernst	6000	-	60	BERNST	590.423.4568	21-MAY-07
105	David	Austin	4800	-	60	DAUSTIN	590.423.4569	25-JUN-05
106	Valli	Pataballa	4800	-	60	VPATABAL	590.423.4560	05-FEB-06
107	Diana	Lorentz	4200	-	60	DLORENTZ	590.423.5567	07-FEB-07
108	Nancy	Greenberg	12008	-	100	NGREENBE	515.124.4569	17-AUG-02
109	Daniel	Faviet	9000	-	100	DFAVIET	515.124.4169	16-AUG-02
110	John	Chen	8200	-	100	JCHEN	515.124.4269	28-SEP-05

↑
NOT NULL constraint
(Primary Key enforces NOT
NULL constraint.)

↑
NOT NULL
constraint

↑
Absence of NOT NULL constraint (Any row
can contain a null value for this column.)

Note: NOT NULL constraints can be created ONLY at the column level.

UNIQUE KEY Constraint

EMPLOYEES

UNIQUE KEY constraint

EMPLOYEE_ID	LAST_NAME	EMAIL
100	King	SKING
101	Kochhar	NKOCHHAR
102	De Haan	LDEHAAN
103	Hunold	AHUNOLD
104	Ernst	BERNST

...



INSERT INTO

208 SMITH	JSMITH
-----------	--------

← Allowed

209 SMITH	JSMITH
-----------	--------

← Not allowed: already exists

UNIQUE Constraint

Defined at either the table level or the column level:

```
CREATE TABLE employees(  
    employee_id      NUMBER(6),  
    last_name        VARCHAR2(25) NOT NULL,  
    email            VARCHAR2(25),  
    salary           NUMBER(8,2),  
    commission_pct   NUMBER(2,2),  
    hire_date        DATE NOT NULL,  
    ...  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

PRIMARY KEY Constraint

DEPARTMENTS

PRIMARY KEY

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500

Not allowed
(null value)



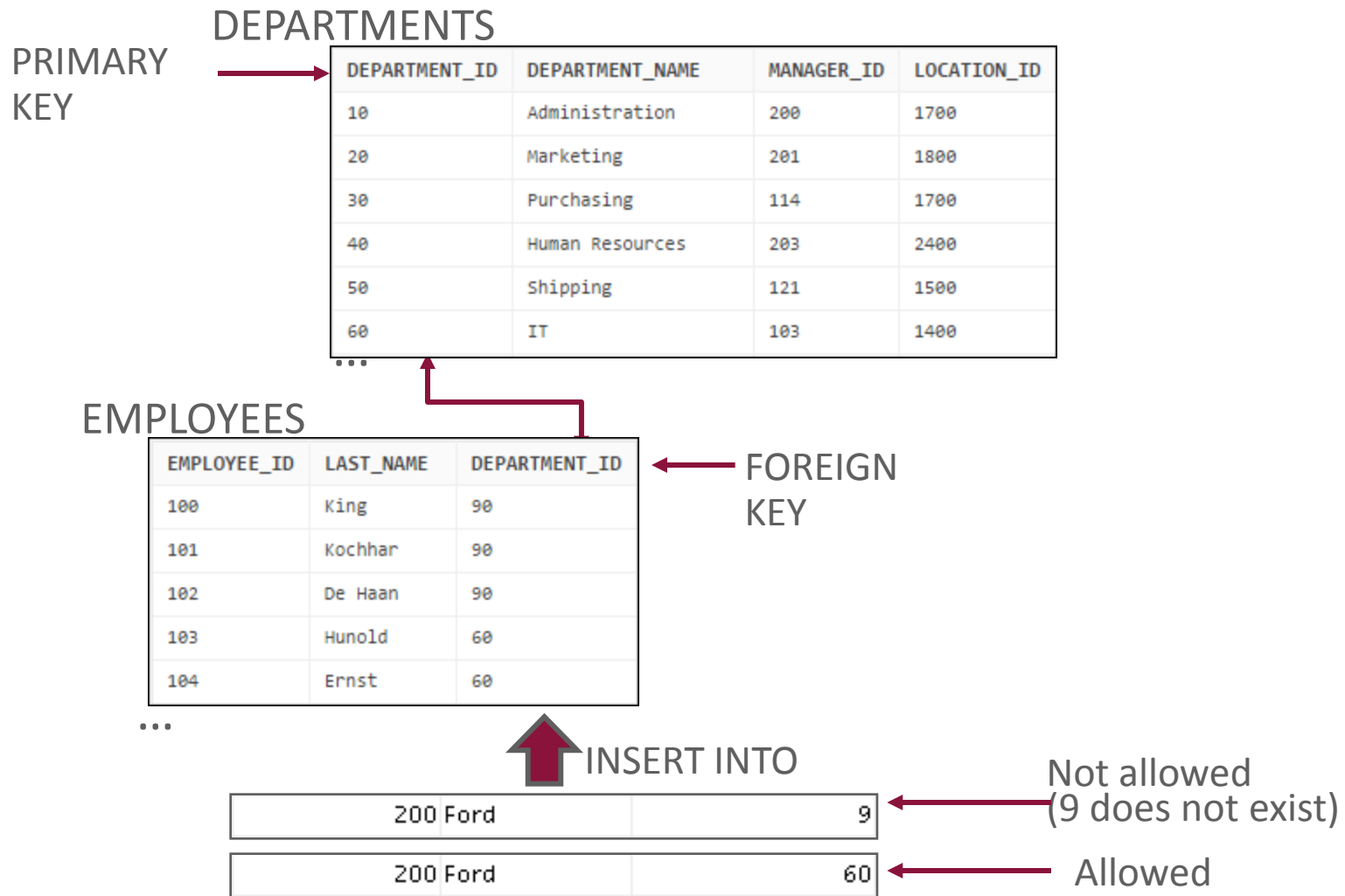
INSERT INTO

(null)	Public Accounting	124	2500
50	Finance	124	1500

Not allowed
(50 already exists)



FOREIGN KEY Constraint



FOREIGN KEY Constraint

Defined at either the table level or the column level:

```
CREATE TABLE employees(  
    employee_id      NUMBER(6),  
    last_name        VARCHAR2(25) NOT NULL,  
    email            VARCHAR2(25),  
    salary           NUMBER(8,2),  
    commission_pct   NUMBER(2,2),  
    hire_date        DATE NOT NULL,  
    ...  
    department_id    NUMBER(4),  
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)  
        REFERENCES departments(department_id),  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

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

CHECK Constraint

- It defines a condition that each row must satisfy.
- It cannot reference columns from other tables.

```
..., salary    NUMBER(8,2)  
CONSTRAINT emp_salary_min  
CHECK (salary > 0),...
```

CREATE TABLE: CHECK Constraint Example

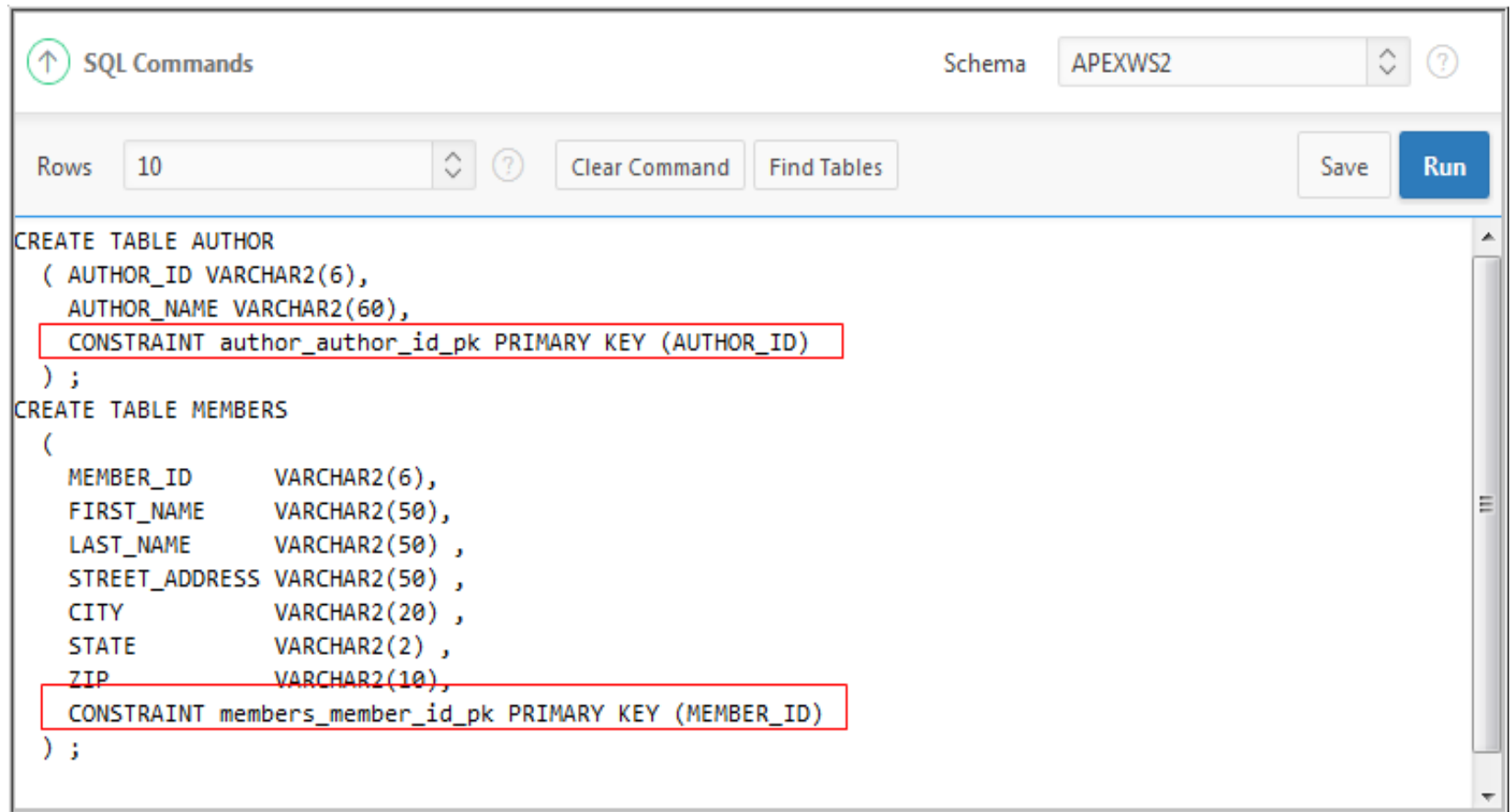
```
CREATE TABLE teach_emp (  
    empno          NUMBER(5) PRIMARY KEY,  
    ename          VARCHAR2(15) NOT NULL,  
    job            VARCHAR2(10),  
    mgr            NUMBER(5),  
    hiredate       DATE DEFAULT (sysdate),  
    photo          BLOB,  
    sal            NUMBER(7,2),  
    deptno         NUMBER(3) NOT NULL  
    CONSTRAINT admin_dept_fkey REFERENCES  
        departments(department_id));
```

Case Scenario: Creating Tables

How about adding the constraints to the simplified library database tables?



Case Scenario: Adding Constraints



The screenshot shows an SQL editor window titled "SQL Commands" with the schema set to "APEXWS2". The interface includes a "Rows" dropdown set to "10", buttons for "Clear Command", "Find Tables", "Save", and "Run". The SQL code in the editor is as follows:

```
CREATE TABLE AUTHOR
(
  AUTHOR_ID VARCHAR2(6),
  AUTHOR_NAME VARCHAR2(60),
  CONSTRAINT author_author_id_pk PRIMARY KEY (AUTHOR_ID)
);
CREATE TABLE MEMBERS
(
  MEMBER_ID      VARCHAR2(6),
  FIRST_NAME     VARCHAR2(50),
  LAST_NAME      VARCHAR2(50),
  STREET_ADDRESS VARCHAR2(50),
  CITY           VARCHAR2(20),
  STATE          VARCHAR2(2),
  ZIP            VARCHAR2(10),
  CONSTRAINT members_member_id_pk PRIMARY KEY (MEMBER_ID)
);
```

Case Scenario: Adding Constraints

```
CREATE TABLE PUBLISHER
(
  PUBLISHER_ID  VARCHAR2(6),
  PUBLISHER_NAME VARCHAR2(100) NOT NULL,
  CONSTRAINT publisher_id_pk PRIMARY KEY (PUBLISHER_ID)
);

CREATE TABLE BOOKS
(
  BOOK_ID      VARCHAR2(6),
  TITLE       VARCHAR2(255) NOT NULL,
  PUBLISHER_ID VARCHAR2(6),
  AUTHOR_ID   VARCHAR2(6),
  CONSTRAINT book_book_id_pk PRIMARY KEY (BOOK_ID),
  CONSTRAINT bk_auth_fk FOREIGN KEY (author_id) REFERENCES author(author_id),
  CONSTRAINT bk_publ_fk FOREIGN KEY (publisher_id) REFERENCES publisher(publisher_id)
);
```

```
CREATE TABLE BOOK_TRANSACTION
(
  TRANSACTION_ID  VARCHAR2(6),
  TRANSACTION_DATE DATE DEFAULT SYSDATE NOT NULL,
  TRANSACTION_TYPE VARCHAR2(10) ,
  BOOK_ID        VARCHAR2(6) ,
  MEMBER_ID      VARCHAR2(6),
  CONSTRAINT booktrns_id_pk PRIMARY KEY (TRANSACTION_ID),
  CONSTRAINT bk_trns_fk FOREIGN KEY (book_id) REFERENCES books(book_id),
  CONSTRAINT bk_mem_fk FOREIGN KEY (member_id) REFERENCES members(member_id)
);
```

ALTER TABLE Statement

Use the `ALTER TABLE` statement to:

- Add a column
- Modify an existing column definition
- Define a default value for the new column
- Drop a column
- Rename a column
- Change a table to read-only status

ALTER TABLE Statement

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

```
ALTER TABLE table
ADD          (column data type [DEFAULT expr]
             [, column data type]...);
```

```
ALTER TABLE table
MODIFY      (column data type [DEFAULT expr]
             [, column data type]...);
```

```
ALTER TABLE table
DROP (column [, column] ...);
```

Adding a Column

- You use the ADD clause to add columns:

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

- The new column becomes the last column:

EMPLOYEE_ID	LAST_NAME	HIRE_DATE	JOB_ID
100	King	17-JUN-03	-
101	Kochhar	21-SEP-05	-
102	De Haan	13-JAN-01	-
103	Hunold	03-JAN-06	-

Modifying a Column

- You can change a column's data type, size, and default value:

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

- A changed default value affects only subsequent insertions in the table.

Dropping a Column

Use the `DROP COLUMN` clause to drop columns that you no longer need:

```
ALTER TABLE dept80  
DROP (job_id);
```

Table altered.

EMPLOYEE_ID	LAST_NAME	anssal	HIRE_DATE
100	King	24000	17-JUN-03
101	Kochhar	17000	21-SEP-05
102	De Haan	17000	13-JAN-01
103	Hunold	9000	03-JAN-06
104	Ernst	6000	21-MAY-07
105	Austin	4800	25-JUN-05
106	Pataballa	4800	05-FEB-06

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.

SET UNUSED Option

You can specify the `ONLINE` keyword to indicate that DML operations on the table will be allowed while marking the column or columns `UNUSED`.

```
ALTER TABLE      <table_name>  
SET      UNUSED(<column_name> [ , <column_name>]);  
OR  
ALTER TABLE  <table_name>  
SET      UNUSED COLUMN <column_name> [ ,  
<column_name>];
```

```
ALTER TABLE <table_name>  
DROP  UNUSED COLUMNS;
```

Case Scenario: Altering Tables



Faculty

Sean, I was reviewing the AUTHORS table and realized that

- The author's email address field is missing.
- The author's name column length needs to be increased.

Can you make these changes?

Sure, I can do it. Because the modification is adding a new column and is increasing the column length, this should not be an issue.



Student

Case Scenario: Altering Tables

```
ALTER TABLE AUTHOR  
ADD EMAIL VARCHAR2(255);  
ALTER TABLE AUTHOR  
MODIFY AUTHOR_NAME VARCHAR2(100);
```

Looks like the table is
modified:

```
table altered  
table altered
```



Read-Only Tables

You can use the ALTER TABLE syntax to:

- Put a table in read-only mode, which prevents DDL or DML changes during table maintenance
- Put the table back into read/write mode

```
ALTER TABLE employees READ ONLY;  
  
-- perform table maintenance and then  
-- return table back to read/write mode  
  
ALTER TABLE employees READ WRITE;
```

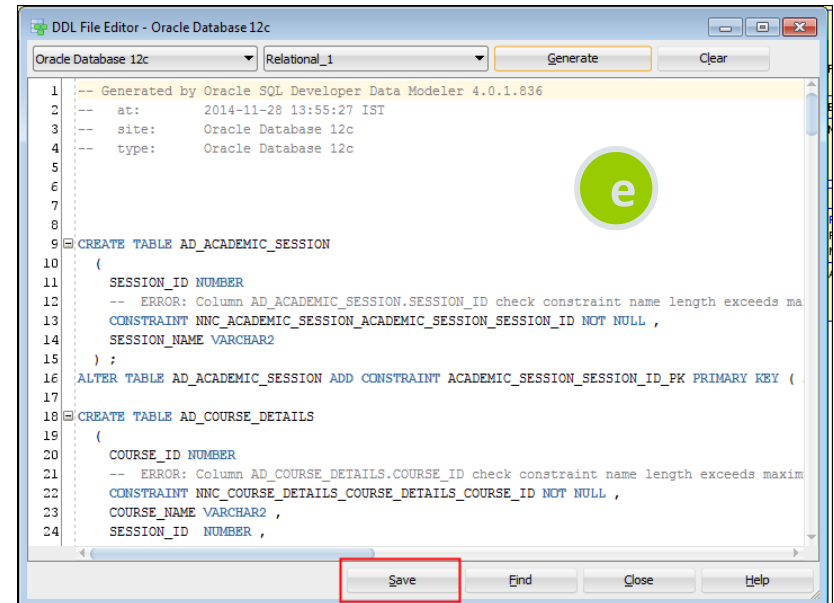
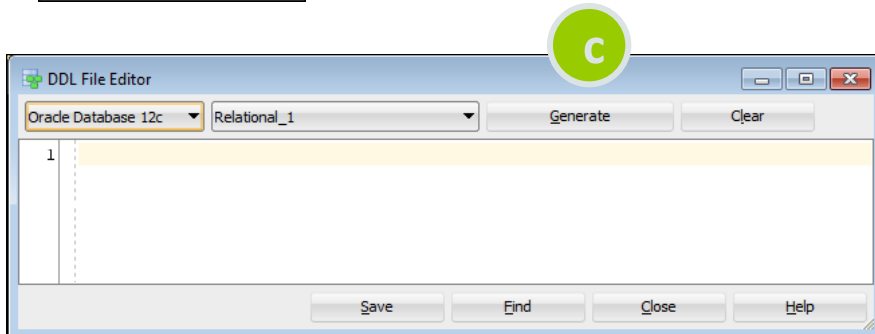
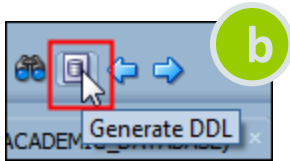
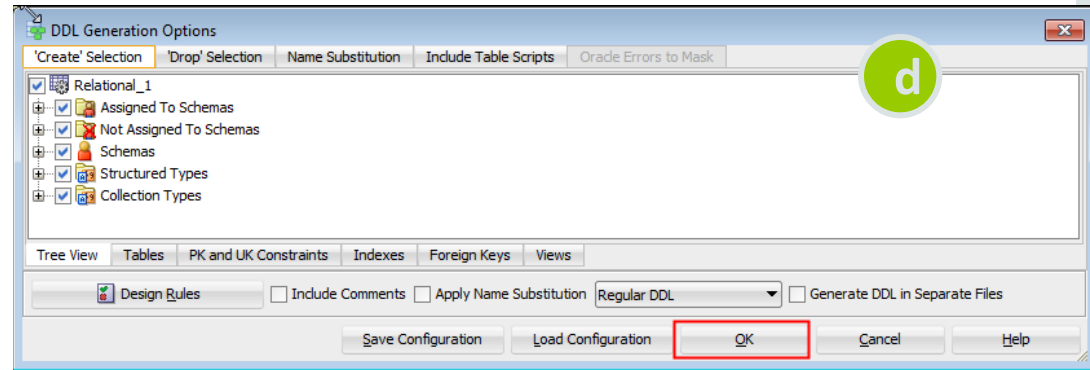
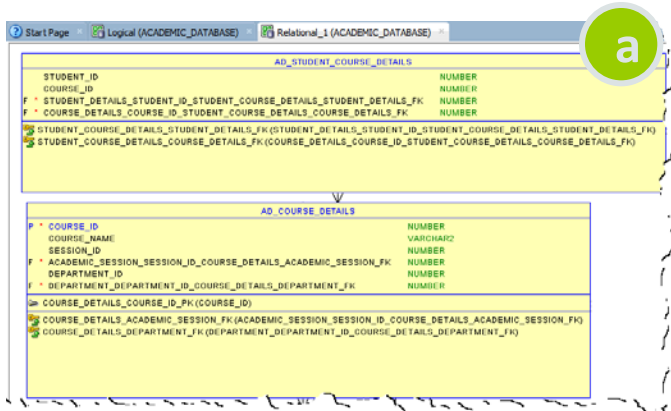
Dropping a Table

- Moves a table to the recycle bin
- Removes the table and its data if the PURGE clause is specified
- Invalidates dependent objects and removes object privileges on the table

```
DROP TABLE dept80;
```

```
Table dropped.
```

Using Oracle SQL Data Modeler to Generate DDL



Summary

In this lesson, you should have learned how to:

- Identify the steps needed to create database tables
- Describe the purpose of the DDL
- List the DDL operations needed to build and maintain a database's tables



