Working with Oracle SQL

Chapter 1:

What Is Structured Query Language?

Chapter Objectives

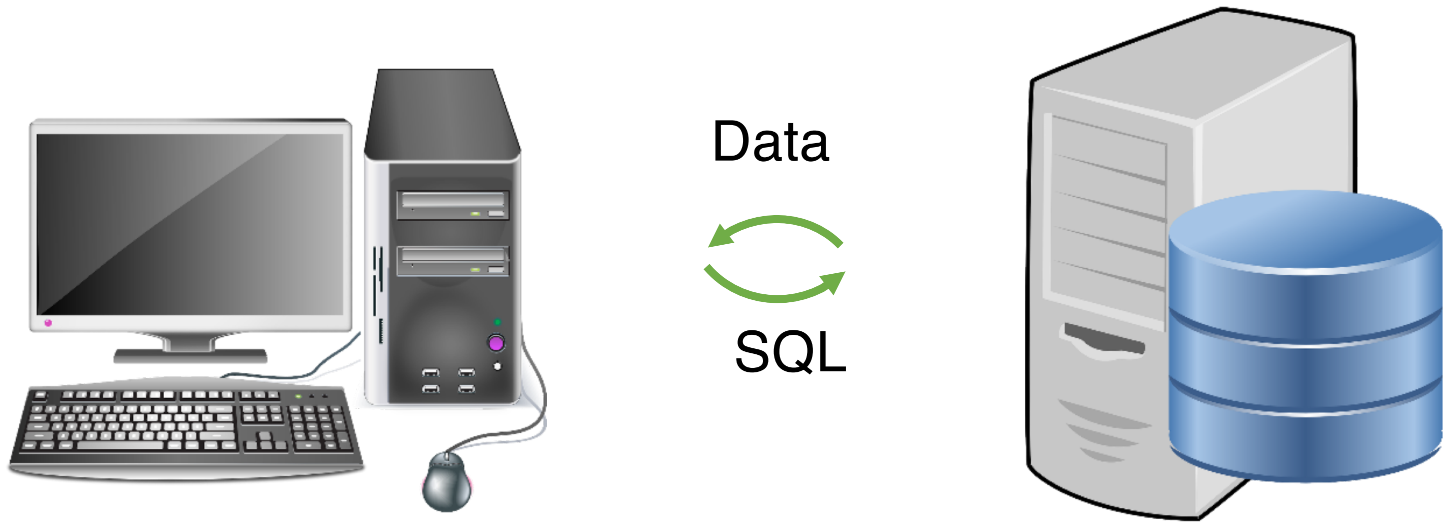
In this chapter, we will discuss:

- The role of SQL
- Concepts of data modeling
- The course environment
- SQL Developer
- SQLPLUS

Chapter Concepts

The Role of SQL

- SQL is a common interface between client and database server
- SQL is the interface between the application program and the database
  - SQL stands for Structured Query Language
  - But, SQL is really a data sublanguage, which is more like an access method than a complete programming language



The Role of SQL (continued)

- Application programs may be:
  - 3GL programs
  - 4GL or application generator programs
  - Report generator programs
  - End-user point-and-click programs
  - Spreadsheets
  - Any frontend tool with SQL interface capability
  - Stored PL/SQL procedures
- Your ability to produce real-world programs will depend on your ability to write SQL statements

Result-Oriented

- SQL is a result–oriented language
  - Specify the desired result rather than step-by-step instructions of what to do
- Example:
  - If we have this table:

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

  - Then this query against the Scott schema:

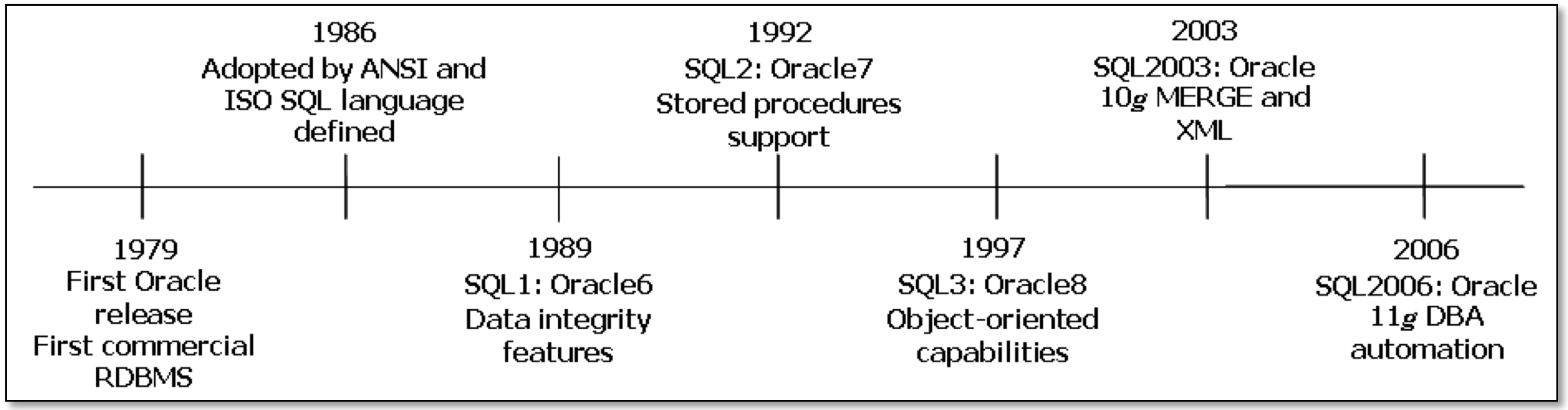
```
SELECT deptno, dname
FROM   dept
WHERE  loc = 'DALLAS';
```

  - Will produce this result:

DEPTNO	DNAME
20	RESEARCH

SQL Standard

- SQL is the standard language for relational databases
  - Unfortunately, different products implement commands differently
  - This course will adhere to the standards where possible
    - Some topics will be non-standard but will be indicated as such
- SQL functionality has evolved significantly over the years



1979
First Oracle release
First commercial RDBMS
1986
Adopted by ANSI and ISO SQL language defined
1989
SQL1: Oracle6
Data integrity features
1992
SQL2: Oracle7
Stored procedures support
1997
SQL3: Oracle8
Object-oriented capabilities
2003
SQL2003: Oracle 10g MERGE and XML
2006
SQL2006: Oracle 11g DBA automation

Chapter Concepts

What Is SQL?

Designing a Database

The Course Environment

Using SQL Developer

Using `sqlplus`

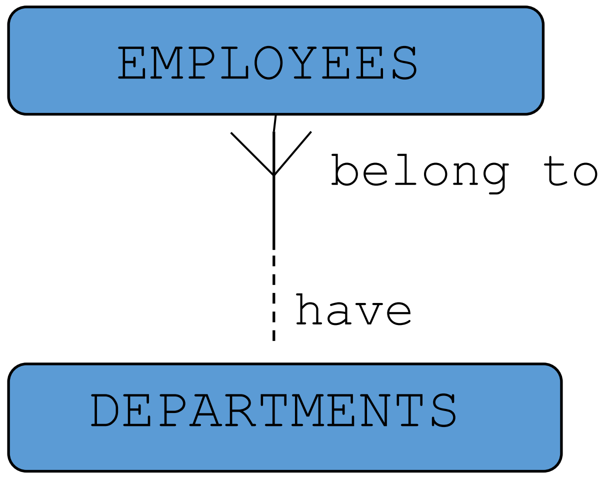
Chapter Summary

Logical Data Model

- Provides a level of abstraction from physical database design by representing data in terms of “logical” or business entities and the relationships between them
- Represents business information and rules
- Provides the input to physical database design
- Comprised of four critical elements
  - Entities
  - Attributes
  - Relationships
  - Candidate keys

Entity

- An object of importance
  - A uniquely identifiable person, place, thing, action, concept, object, or event about which information needs to be known or held
- Represented as a soft box
- Example:



Attribute

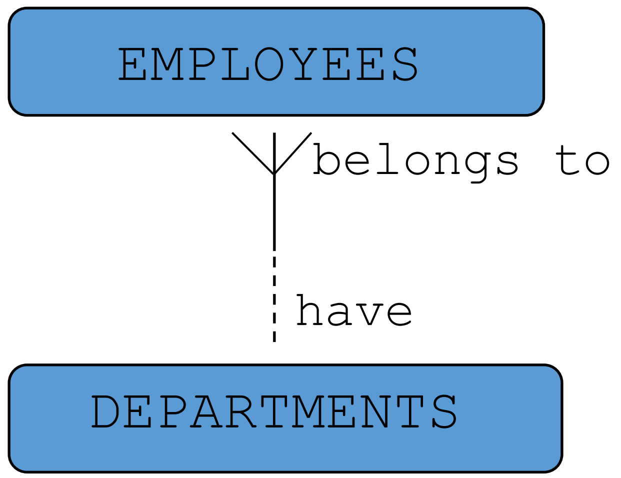
- A fact that is a nondecomposable unit of information about an entity
  - Qualify, identify, classify, quantify, or express the state of an entity
- Example:
  - `employee_id`, `last_name`, and `first_name` are attributes of the `employees` entity
- Further defined by indicating whether or not it is mandatory
  - That is, if it must exist for every occurrence of an entity
- Example:
  - `employee_id` is mandatory, whereas `phone_number` is not required

Attribute Datatype

- Each attribute is further qualified by a datatype
- Common datatypes are `NUMBER`, `CHAR`, and `DATE`
    - `NUMBER` represents numerical values, `CHAR` represents character strings, and `DATE` represents dates

Relationship

- Association between two entities
- Defined by a verb or a preposition connecting two entities
- Both ends must be named
- Example:



Relationship Cardinality

- Cardinality defines the expected number of related occurrences for each entity
- Most common cardinality is one to many (1:M)
- Example:
  - Department may have many employees, while each employee must represent one and only one department
- Indicates the parent entity (at the “one” end) and the child entity (at the “many” end)
- Example:
  - Departments is the parent and employees is the child

Relationship Optionality

- Defines coexistence of the two entities
- Defined at both ends of the relationship to indicate if a parent can exist without a child and if the child can exist without a parent
- Example:
    - Department may or may not have employees, whereas an employee must belong to a department

Reading Relationships

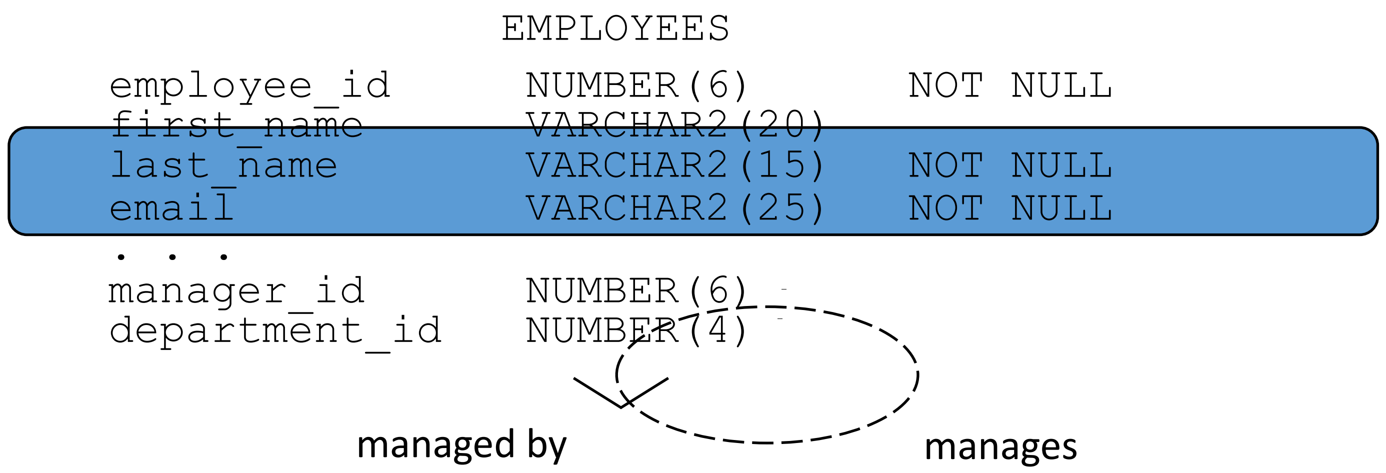
- Relationships are read in both directions
  - Cardinality and optionality are both included

EACH	**Entity1**	MAY MUST	**Relationship**	ONE AND ONLY ONE ONE OR MORE	**Entity2**

- Examples:
  - Each `department` may `have` one or more `employees`
  - Each `employee` must `belong to` one and only one `department`

Recursive Relationship

- A relationship from an entity onto itself
- Captures a hierarchical structure, such as a reporting tree in an organization
- For example, a recursive relationship captures the fact that each employee may be managed by another employee
  - Each employee may manage one or more employees
  - Each employee may be managed by one and only one employee



```
                         EMPLOYEES
employee_id       NUMBER(6)        NOT NULL
first_name        VARCHAR2(20)
last_name         VARCHAR2(15)     NOT NULL
email             VARCHAR2(25)     NOT NULL
. . .
manager_id        NUMBER(6)
department_id     NUMBER(4)
```

Candidate Key

- An attribute or a minimal set of attributes that uniquely identify a specific row
- Examples:
  - `employee_id` uniquely identifies a employee
  - A combination of `first_name`, `last_name`, and `phone_number` uniquely identify an employee

Transforming a Logical Data Model to a Database Design

- Concepts in the logical model are mapped to database structures

Logical	Physical
Entity	Table
Attribute	Column
Candidate key	Primary or unique key
Relationship	Foreign key

- Tables and columns are usually a simple mapping from entities and attributes

Primary and Unique Keys

- A *unique key* has the same definition as a candidate key: a column or a minimum set of columns that uniquely identifies a specific row
- A *primary key* has the same definition as a unique key but with two further restrictions
  - It must be composed of mandatory columns
  - Only one primary key is allowed for each table
- Once the primary key is selected from a valid list of candidate keys, the remaining candidate keys are mapped to unique keys
- Example:
  - `employee_id` will be selected as a primary key because it is mandatory
  - The combination of `first_name`, `last_name`, and `phone_number` will become a unique key

Foreign Key

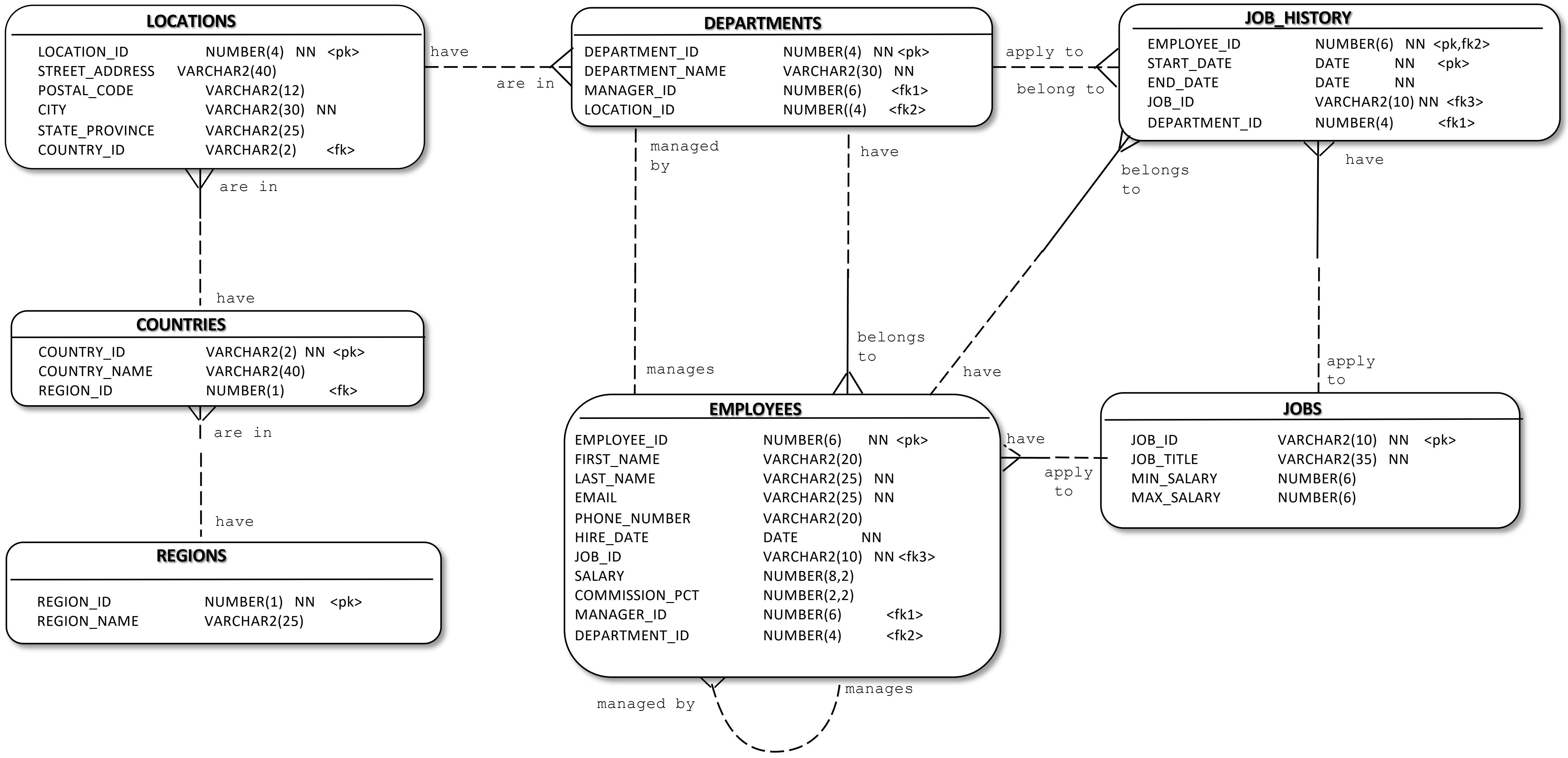
- By definition, a relationship copies candidate key columns of a parent table to a child table
- *Foreign key* enforces this relationship using two rules
    - Values in the relationship columns of the child table exist in the parent table
    - Cannot change values in the parent table that are referenced in the child table
- Example:
    - Foreign key corresponding to the department – employee relationship has these rules
        - The value of `department_id` in the `employee` table must previously exist in the `departments` table
        - The value of `department_id` in the `departments` table cannot be modified if it is used in the `employees` table

Chapter Concepts

Course Exercises

- The exercises in this course all use standard Oracle user accounts
- When an Oracle Database is created, Oracle optionally embeds several users with populated schemas
    - These can then be used for training and demonstrating various features
- For experience in using multiple accounts, two of these user accounts (`HR` and `SCOTT`) have been selected for use
    - In each exercise, be careful to connect to the appropriate account
- The implication is that the students of this course can then go back and practice the lab exercises at any time without needing special setups from ROI

ER Diagram—HR Account



LOCATIONS

LOCATION_ID	NUMBER(4)	NN	<pk>
STREET_ADDRESS	VARCHAR2(40)		
POSTAL_CODE	VARCHAR2(12)		
CITY	VARCHAR2(30)	NN	
STATE_PROVINCE	VARCHAR2(25)		
COUNTRY_ID	VARCHAR2(2)		<fk>

DEPARTMENTS

DEPARTMENT_ID	NUMBER(4)	NN	<pk>
DEPARTMENT_NAME	VARCHAR2(30)	NN	
MANAGER_ID	NUMBER(6)		<fk1>
LOCATION_ID	NUMBER((4)		<fk2>

JOB_HISTORY

EMPLOYEE_ID	NUMBER(6)	NN	<pk,fk2>
START_DATE	DATE	NN	<pk>
END_DATE	DATE	NN	
JOB_ID	VARCHAR2(10)	NN	<fk3>
DEPARTMENT_ID	NUMBER(4)		<fk1>

COUNTRIES

COUNTRY_ID	VARCHAR2(2)	NN	<pk>
COUNTRY_NAME	VARCHAR2(40)		
REGION_ID	NUMBER(1)		<fk>

EMPLOYEES

EMPLOYEE_ID	NUMBER(6)	NN	<pk>
FIRST_NAME	VARCHAR2(20)		
LAST_NAME	VARCHAR2(25)	NN	
EMAIL	VARCHAR2(25)	NN	
PHONE_NUMBER	VARCHAR2(20)		
HIRE_DATE	DATE	NN	
JOB_ID	VARCHAR2(10)	NN	<fk3>
SALARY	NUMBER(8,2)		
COMMISSION_PCT	NUMBER(2,2)		
MANAGER_ID	NUMBER(6)		<fk1>
DEPARTMENT_ID	NUMBER(4)		<fk2>

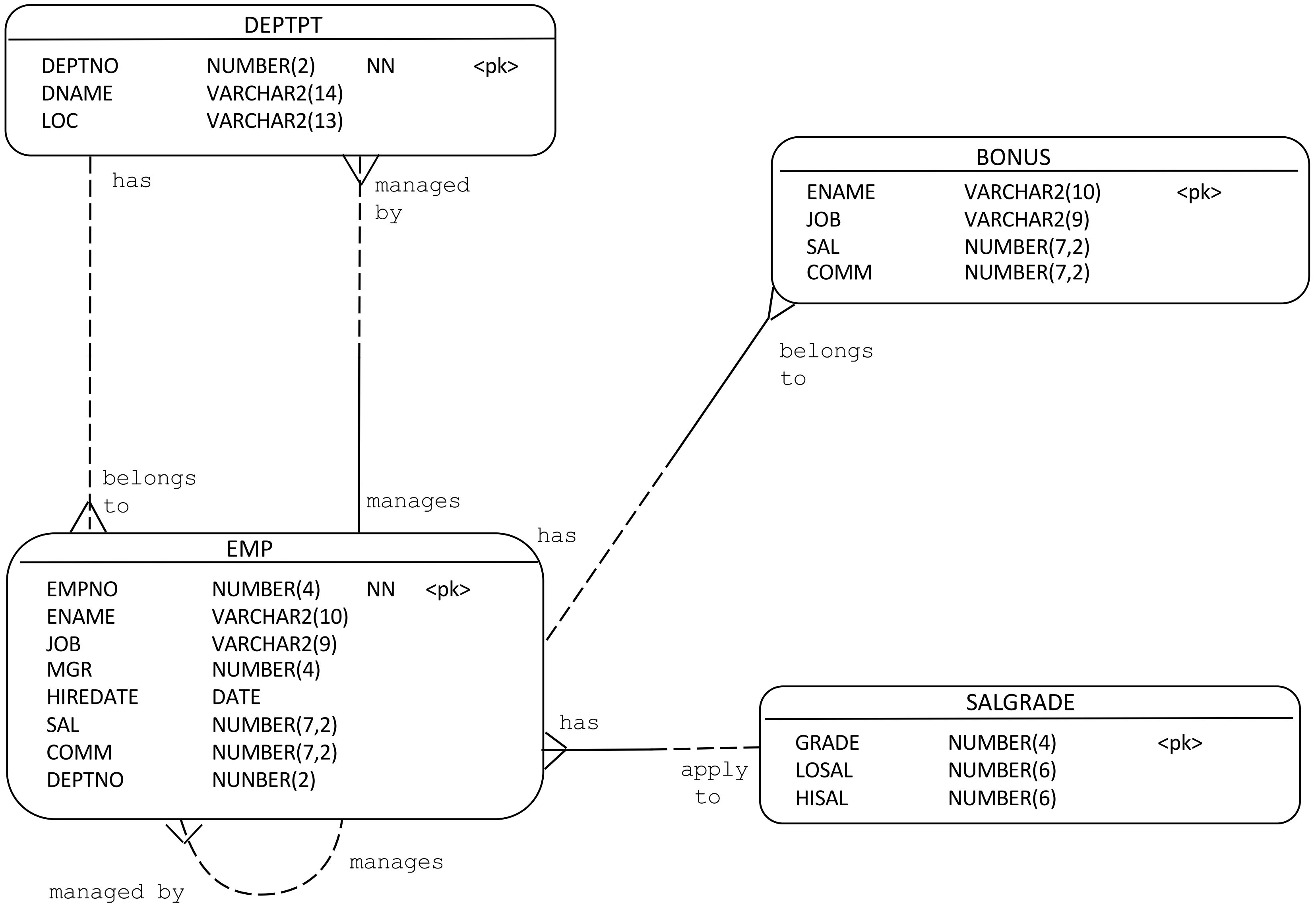
JOBS

JOB_ID	VARCHAR2(10)	NN	<pk>
JOB_TITLE	VARCHAR2(35)	NN	
MIN_SALARY	NUMBER(6)		
MAX_SALARY	NUMBER(6)		

REGIONS

REGION_ID	NUMBER(1)	NN	<pk>
REGION_NAME	VARCHAR2(25)		

ER Diagram—SCOTT Account



Conventions for Command Syntax

- The following command syntax is used in this course:

Feature	Example	Explanation
Uppercase	CREATE	Reserved word; enter exactly as spelled
Lowercase	column_name	Substitute an appropriate value
Three periods	role_name,..., role_name	Items may be repeated any number of times
Square brackets	[NOT NULL]	Optional item
Vertical bar	ON | OFF	Alternative item; use one or the other

Chapter Concepts

What Is SQL?

Designing a Database

The Course Environment

Using SQL Developer

Using `sqlplus`

Chapter Summary

SQL Developer

- Oracle SQL Developer is a graphical tool used to work with a database
  - Simplifies basic tasks for DBAs and developers
  - Released in 2006
- Developed in Java
  - Runs on Windows, Linux, and Mac OS X
- Supports Oracle 9i and later
- Key concepts
  - Connections
  - Object Navigator
  - SQL Worksheet

Connections

- Each connection is configured for a single Oracle user
  - Uses standard Oracle database authentication
- Can also connect to third party databases
  - Access, SQL Server, MySQL
- All connections are listed in the Connections window
  - Drill down each connection to view the objects to which the user has access
- Create a new connection using the icon (+) at the top of the Connections window
  - Requires information about the server, such as user, password, server name

Object Navigator and Details

- Expanding a Connection node exposes the Object Navigator
  - Automatically connects if not connected yet
- First level under the connection is a list of object types such as tables, views, and indexes
- Next level contains a list of objects
  - Example: table names (`departments`, `employees`, ...)
- When an object is selected, specific information is displayed
  - Information varies by object type
  - Selected by clicking an object in the Object Navigator
- Tables have the following commonly used tabs:
  - Columns—displays the structure of the table (columns, datatypes, etc.)
  - Data—displays the data from the table
  - SQL—includes the SQL to create the object

SQL Worksheet

- SQL Worksheet allows you to enter SQL and PL/SQL statements
    - Also supports some SQL*Plus commands
- Top window is a SQL statement editor
    - Supports both DML and DDL statements
    - Examples: creating tables, inserting data, selecting data
- Bottom windows display results
- Key components
    - Editor
    - Results window
    - Script Output window

Editor

- Used for writing SQL statements and executing scripts
  - Oracle keywords are highlighted automatically
  - Supports standard file operations such as open, save, and print
  - The Eraser icon clears the contents of the Editor
- To pull in a column or a table, drag it from the Object Navigator
  - For tables, a SQL statement is created automatically
- To format the statement, right-click in the Editor and select Format SQL
- To recall a previous command, click SQL History icon
  - History is maintained even if you close SQL Developer

Results Window

- Displays output from a single `SELECT` statement
    - Execute using Execute Statement icon or `<F9>` function key
- If multiple statements exist in the editor, only the one where the cursor is located will be executed
    - The line where the cursor exists is highlighted
    - Each statement must end with a semicolon, otherwise an error is displayed
- To sort data by one column, double-click the column heading
    - Once for ascending, a second time for descending
    - To sort by multiple columns, use an `ORDER BY` clause
- Right-click in the Results window to use the following features:
    - Auto Fit to format column widths
    - Count Rows to get the total number of records returned
    - Single Record View to view a single record at a time

Script Output Window

- Displays results of all commands in the editor
  - Execute using Run Script icon or `<F5>` function key
- Each statement in the editor is executed one after another
- For each statement, all results are displayed one after another
  - In contrast, the Results Window displays results of only one statement and only 50 rows at a time
  - Number of rows can be changed in Preferences Worksheet Parameters
- Use icons at the top of the Script Output Window to:
  - Clear the contents of the window
  - Save the contents to a file
  - Print the contents

Exporting Data

- Data can be exported from the Results Window and the Table Data Tab
    - Right-click in the Results Window and select Export Data
- Export Data pop-up provides the following features:
    - Output to a File or the Clipboard
    - Choose a file format such as TEXT, CSV, XML, HTML, XLS
    - Select the columns to include:
        - Default is ALL
    - Enter a `WHERE` clause to restrict the results
- The Apply button generates the file

Exercise 1.1: Using SQL Developer

HANDS-ON EXERCISE

20 min

- Please complete this exercise in your Exercise Manual

Chapter Concepts

What Is SQL?

Designing a Database

The Course Environment

Using SQL Developer

➤ **Using `sqlplus`**

Chapter Summary

sqlplus

- SQL*Plus is primarily a command-line application, but, despite its lack of “flash,” it is a workhorse tool used daily by database administrators, developers, and yes, even end users.

- SQL*Plus is essentially an interactive query tool with some scripting capabilities.
  - You can enter a SQL statement, such as a SELECT query, and view the results.
  - You can execute *data definition language* (DDL) statements to create tables and other objects.
  - DBAs can use SQL*Plus to start up, shut down, and otherwise administer a database.
  - You can even enter and execute PL/SQL code.

Chapter Concepts

Chapter Summary

In this chapter, we have discussed:

- The role of SQL
- Concepts of data modeling
- The course environment
- SQL Developer
- The Fidelity Development Environment