SE212 Database System and Database design

Agenda

- Term Project (953212 + Interactive web pro)
- Intro SQL

Midterm score 30%

- Still grading 4/34
- Drop with W date

Term Project

- 2 people only for 1 team
- https://docs.google.com/document/d/1-ZWkBvLol0Pm1PmwlNskjp5cDoE1q5gxe26oK6lABnY/edit?usp=sharing
- Proposal due this Sunday (5%) (15%)
  - Introduction/Background
  - Motivation
  - Business rules
  - Framework
  - ER, EER diagram
  - Schema
  - SDLC
  - Normalization and Technologies involved
  - UI/Forms Mockup (CRUD)

Project due date

- Backend (DB) + simple form
- Due after the final exam of this class
- Frontend (with framework like Angular7)
- Due after the final exam of that class

Objectives

- Definition of terms
- Interpret history and role of SQL
- Define a database using SQL data definition language
- Write single table queries using SQL
- Establish referential integrity using SQL

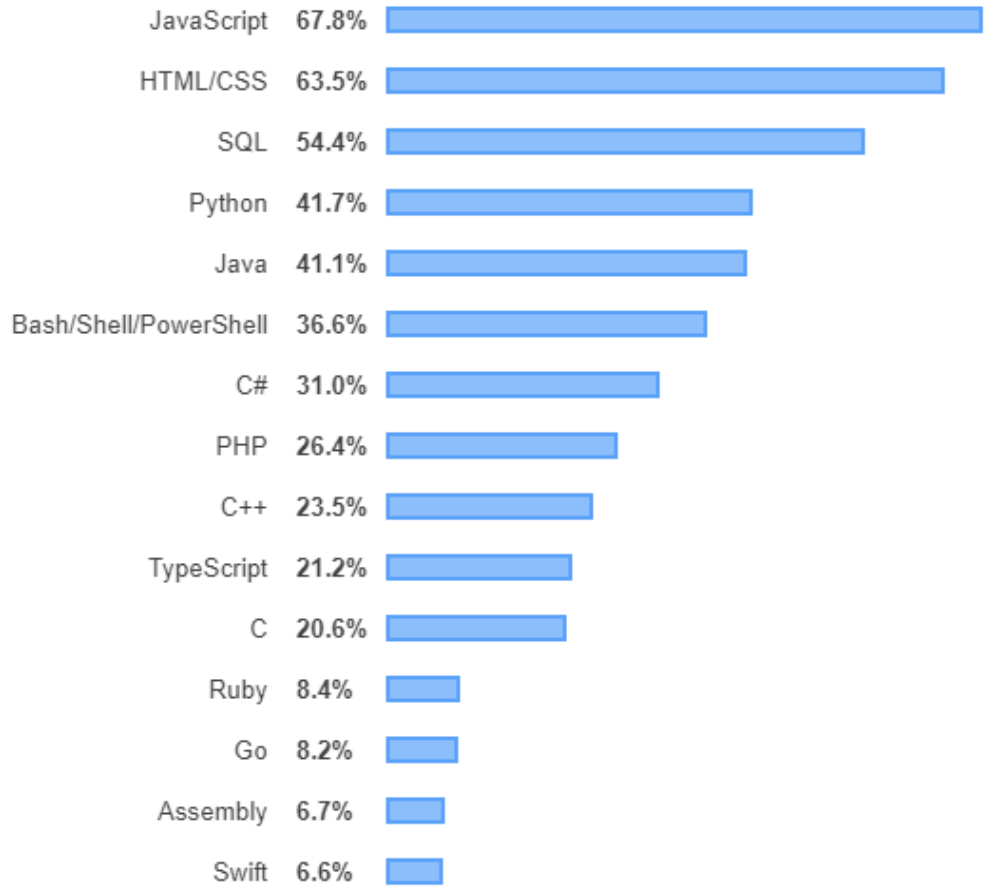
Why SQL?

- 1. SQL is used everywhere
- 2. Its in high demand, many companies use it

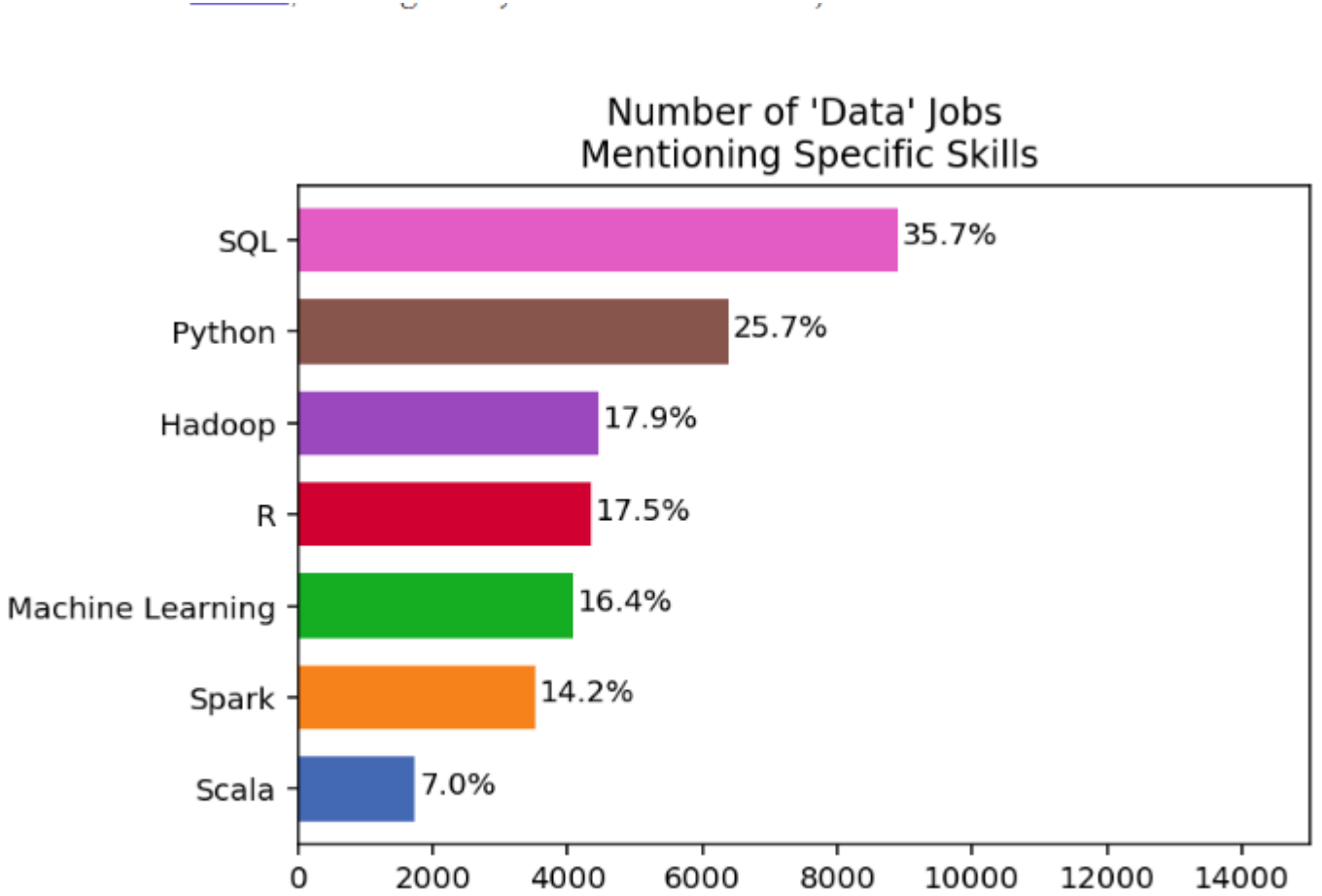
Most Popular Technologies

Programming, Scripting, and Markup Languages

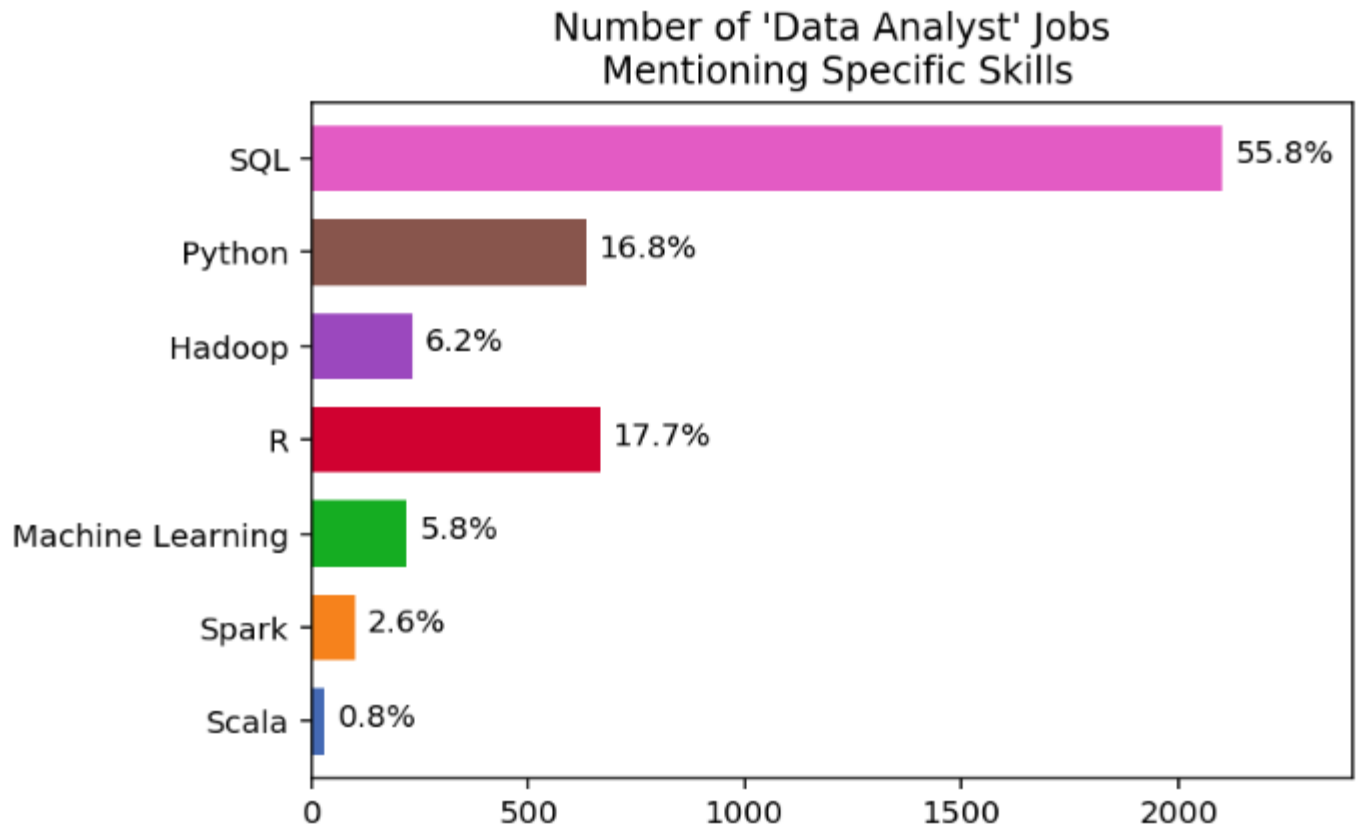
All Respondents | Professional Developers



SQL is in demand



SQL is in demand

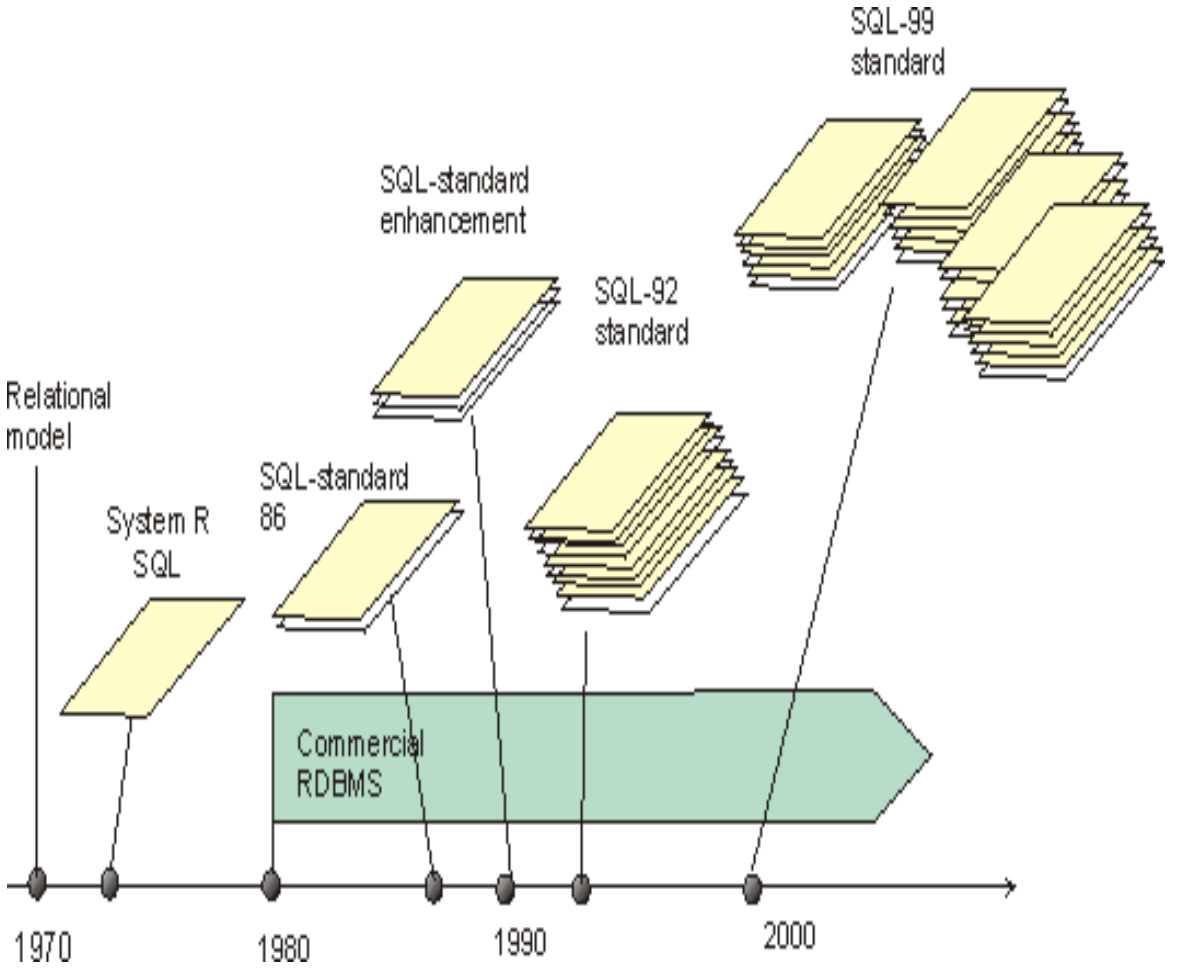


SQL Overview

- Structured Query Language
- The standard for relational database management systems (RDBMS)
- RDBMS: A database management system that manages data as a collection of tables in which all relationships are represented by common values in related tables

History of SQL

- 1970–E. Codd develops relational database concept
- 1974-1979–System R with Sequel (later SQL) created at IBM Research Lab
- 1979–Oracle markets first relational DB with SQL
- 1986–ANSI SQL standard released
- 1989, 1992, 1999, 2003–Major ANSI standard updates
- Current–SQL is supported by most major database vendors



Purpose of SQL Standard

- Specify syntax/semantics for data definition and manipulation
- Define data structures
- Enable portability
- Allow for later growth/enhancement to standard

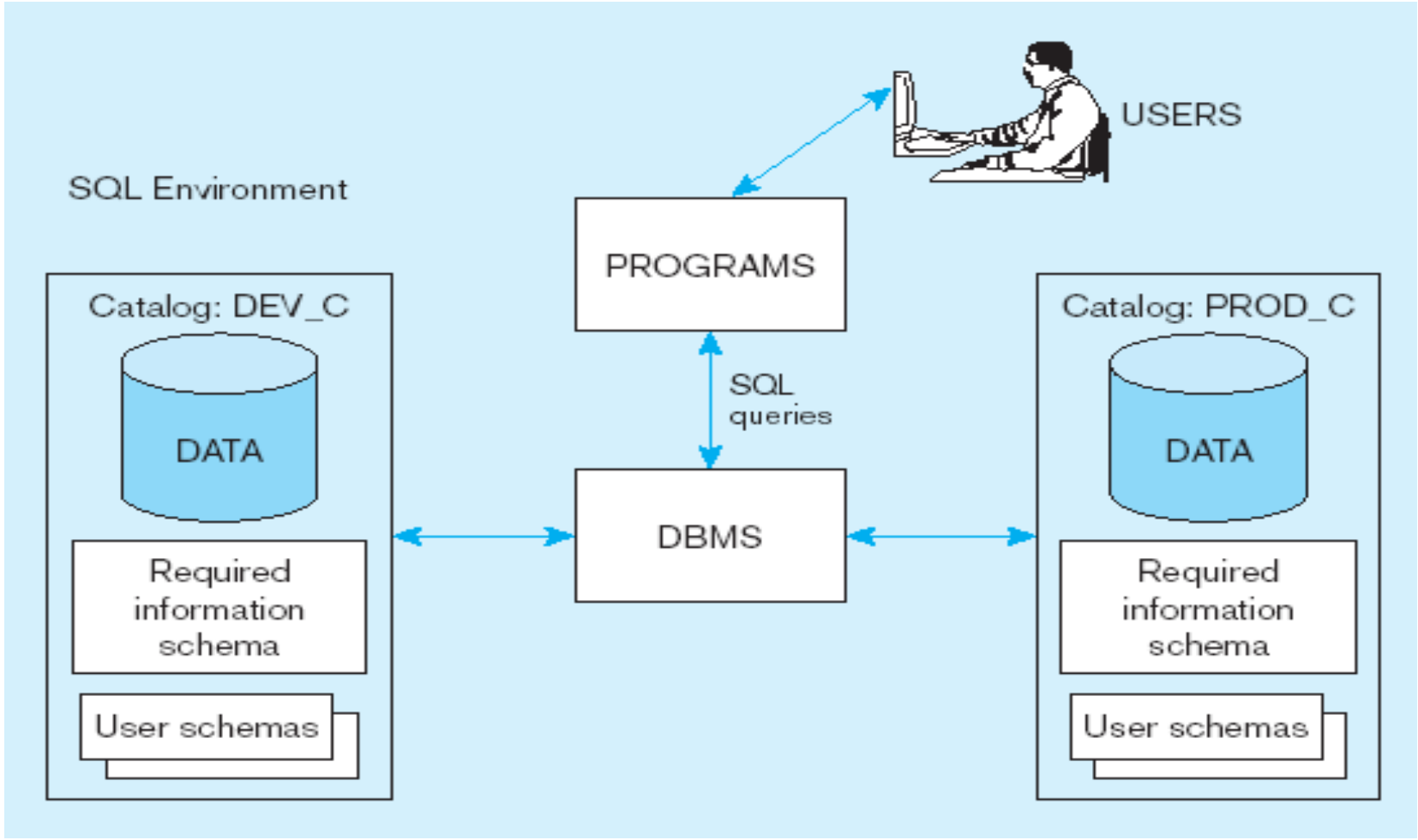
Benefits of a Standardized Relational Language

- Reduced training costs
- Productivity
- Application portability
- Application longevity
- Reduced dependence on a single vendor
- Cross-system communication

SQL Environment

- Catalog (view)
  - A set of schemas that constitute the description of a database
- Schema
  - The structure that contains descriptions of objects created by a user (base tables, views, constraints)
- Data Definition Language (DDL)
  - Commands that define a database, including creating, altering, and dropping tables and **establishing constraints**
- Data Manipulation Language (DML)
  - Commands that maintain and query a database
- Data Control Language (DCL)
  - Commands that control a database, including administering privileges and committing data

A simplified schematic of a typical SQL environment, as described by the SQL-2003 standard

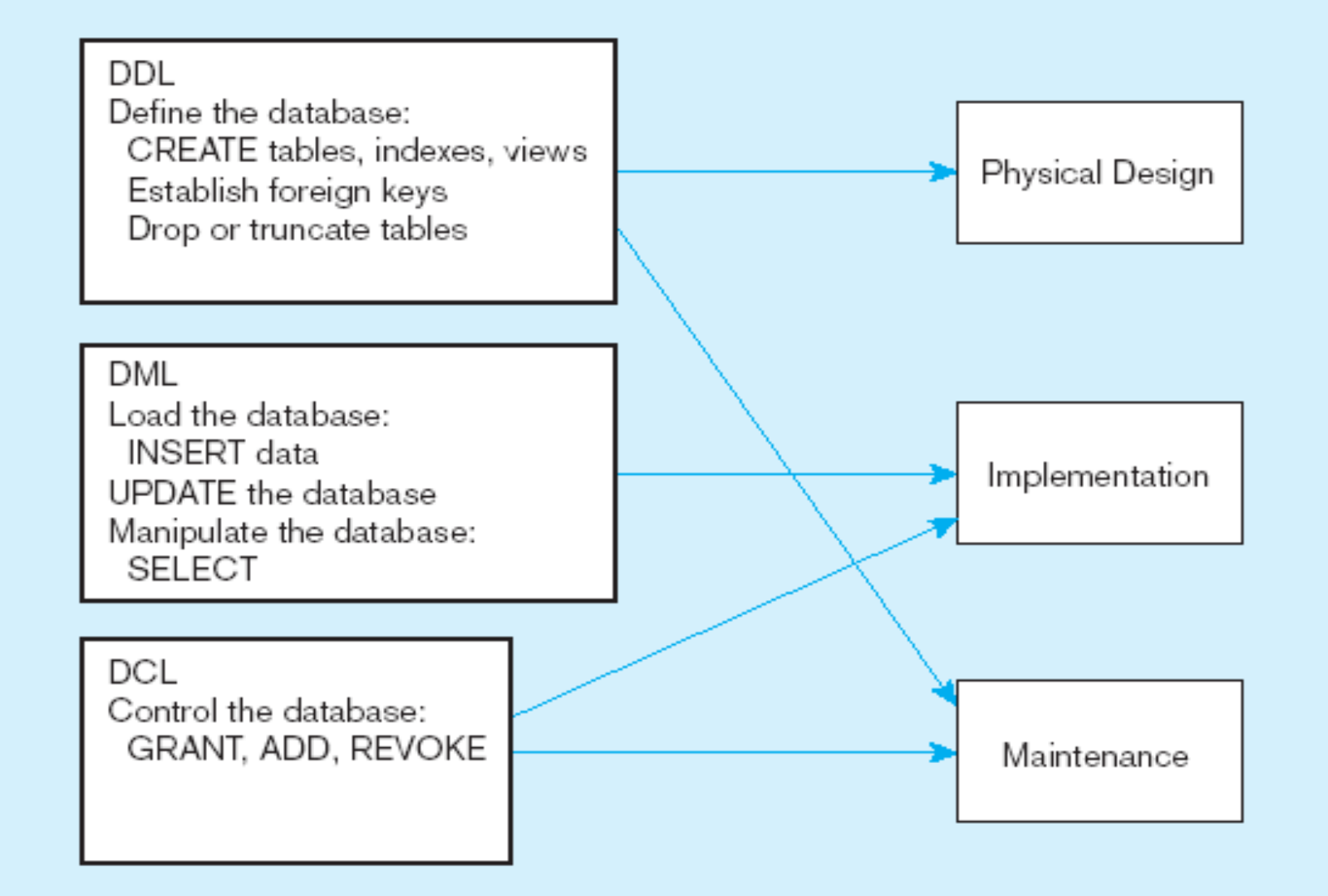


Some SQL Data types

Table 7-2 Sample SQL Data Types

String	CHARACTER (CHAR)	Stores string values containing any characters in a character set. CHAR is defined to be a fixed length.
	CHARACTER VARYING (VARCHAR)	Stores string values containing any characters in a character set, but of definable variable length.
	BINARY LARGE OBJECT (BLOB)	Stores binary string values in hexadecimal format. BLOB is defined to be a variable length.
Number	NUMERIC	Stores exact numbers with a defined precision and scale.
	INTEGER (INT)	Stores exact numbers with a predefined precision and scale of zero.
Temporal	TIMESTAMP	Stores a moment an event occurs, using a definable fraction of a second precision.
Boolean	BOOLEAN	Stores truth values, TRUE, FALSE, or UNKNOWN.

Figure 7-4
DDL, DML, DCL, and the database development process



SQL Database Definition

- Data Definition Language (DDL)
- Major CREATE statements:
  - CREATE SCHEMA–defines a portion of the database owned by a particular user
  - CREATE TABLE–defines a table and its columns
  - CREATE VIEW–defines a logical table from one or more views
- Other CREATE statements: CHARACTER SET, COLLATION, TRANSLATION, ASSERTION, DOMAIN

Table Creation

Figure 7-5 General syntax for CREATE TABLE

```
CREATE TABLE tablename
( {column definition   [table constraint] } . , . .
[ON COMMIT {DELETE | PRESERVE} ROWS] );

where column definition ::=
column_name
        {domain name | datatype [(size)] }
        [column_constraint_clause . . .]
        [default value]
        [collate clause]

and table constraint ::=
        [CONSTRAINT constraint_name]
        Constraint_type [constraint_attributes]
```

Steps in table creation:

1. Identify data types for attributes
2. Identify columns that can and cannot be null
3. Identify columns that must be unique (candidate keys)
4. Identify primary key–foreign key mates
5. Determine default values
6. Identify constraints on columns (domain specifications)
7. Create the table and associated indexes

The following slides create tables for this enterprise data model

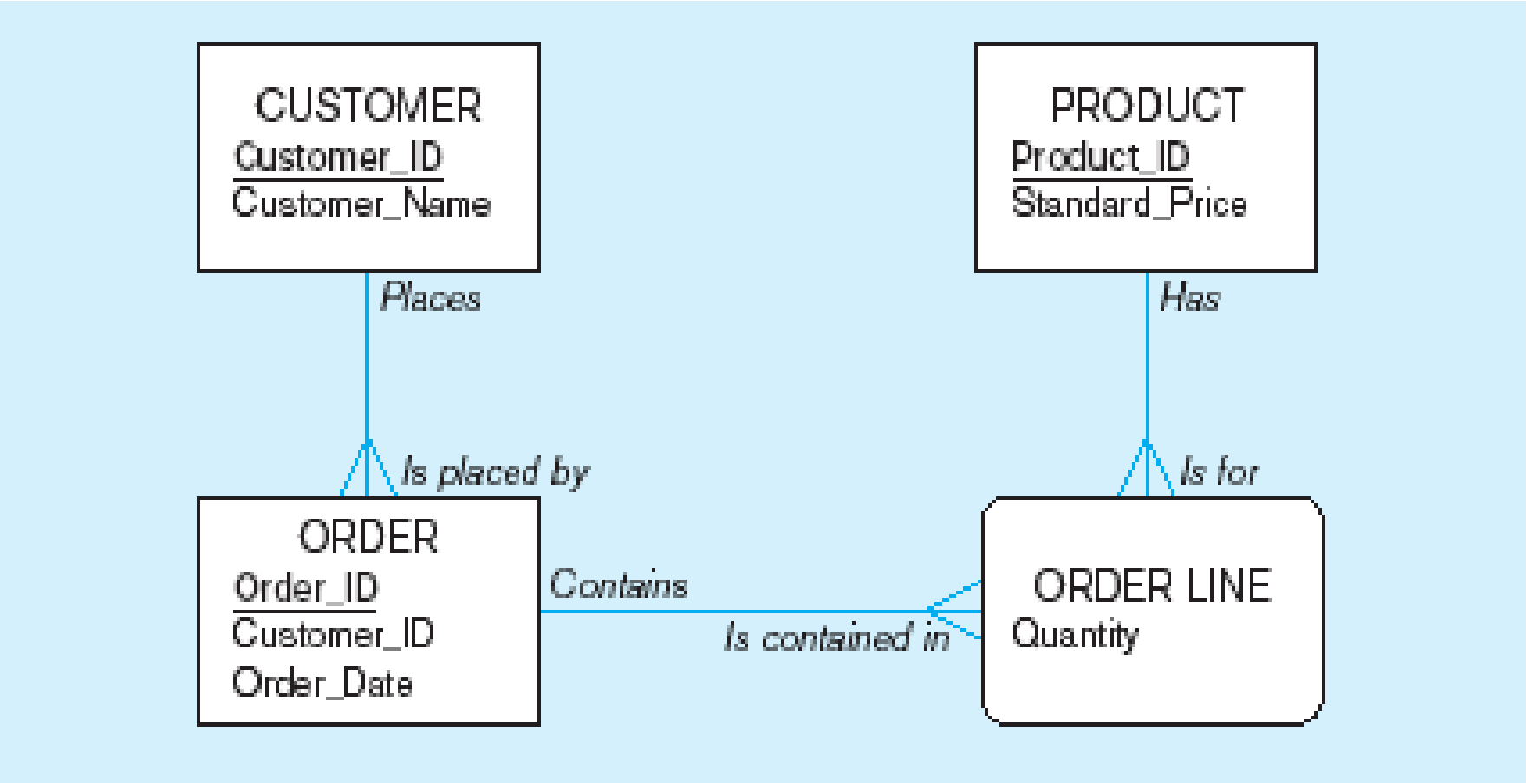


Figure 7-6 SQL database definition commands for Pine Valley Furniture

```
CREATE TABLE CUSTOMER_T
              (CUSTOMER_ID                  NUMBER(11, 0) NOT NULL,
               CUSTOMER_NAME                VARCHAR2(25)  NOT NULL,
               CUSTOMER_ADDRESS             VARCHAR2(30),
               CITY                         VARCHAR2(20),
               STATE                        VARCHAR2(2),
               POSTAL_CODE                  VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
              (ORDER_ID                     NUMBER(11, 0) NOT NULL,
               ORDER_DATE                   DATE DEFAULT SYSDATE,
               CUSTOMER_ID                  NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));

CREATE TABLE PRODUCT_T
              (PRODUCT_ID                   INTEGER        NOT NULL,
               PRODUCT_DESCRIPTION          VARCHAR2(50),
               PRODUCT_FINISH               VARCHAR2(20)
                        CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                'Red Oak', 'Natural Oak', 'Walnut')),
               STANDARD_PRICE               DECIMAL(6,2),
               PRODUCT_LINE_ID              INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
              (ORDER_ID                     NUMBER(11,0)  NOT NULL,
               PRODUCT_ID                   NUMBER(11,0)  NOT NULL,
               ORDERED_QUANTITY             NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

Overall table definitions

Defining attributes and their data types

```
CREATE TABLE PRODUCT_T
        (PRODUCT_ID             INTEGER      NOT NULL,
         PRODUCT_DESCRIPTION    VARCHAR2(50),
         PRODUCT_FINISH         VARCHAR2(20)
                    CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                'Red Oak', 'Natural Oak', 'Walnut')),
         STANDARD_PRICE         DECIMAL(6,2)
         PRODUCT_LINE_ID        INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));
```

```
CREATE TABLE PRODUCT_T
        (PRODUCT_ID             INTEGER     NOT NULL,
         PRODUCT_DESCRIPTION    VARCHAR2(50),
         PRODUCT_FINISH         VARCHAR2(20)
                    CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                'Red Oak', 'Natural Oak', 'Walnut')),
         STANDARD_PRICE         DECIMAL(6,2),
         PRODUCT_LINE_ID        INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));
```

Non-nullable specification

Identifying primary key

Primary keys can never have NULL values

```
CREATE TABLE ORDER_LINE_T
        (ORDER_ID           NUMBER(11,0) NOT NULL,
         PRODUCT_ID         NUMBER(11,0) NOT NULL,
         ORDERED_QUANTITY   NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

Non-nullable specifications

Primary key

Some primary keys are composite–
composed of multiple attributes

Controlling the values in attributes

```
CREATE TABLE ORDER_T
        (ORDER_ID           NUMBER(11, 0) NOT NULL,
         ORDER_DATE         DATE          DEFAULT SYSDATE,
         CUSTOMER_ID        NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));

CREATE TABLE PRODUCT_T
        (PRODUCT_ID         INTEGER       NOT NULL,
         PRODUCT_DESCRIPTION VARCHAR2(50),
         PRODUCT_FINISH     VARCHAR2(20)
                CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                        'Red Oak', 'Natural Oak', 'Walnut')),
         STANDARD_PRICE     DECIMAL(6,2),
         PRODUCT_LINE_ID    INTEGER,
```

Default value

Domain constraint

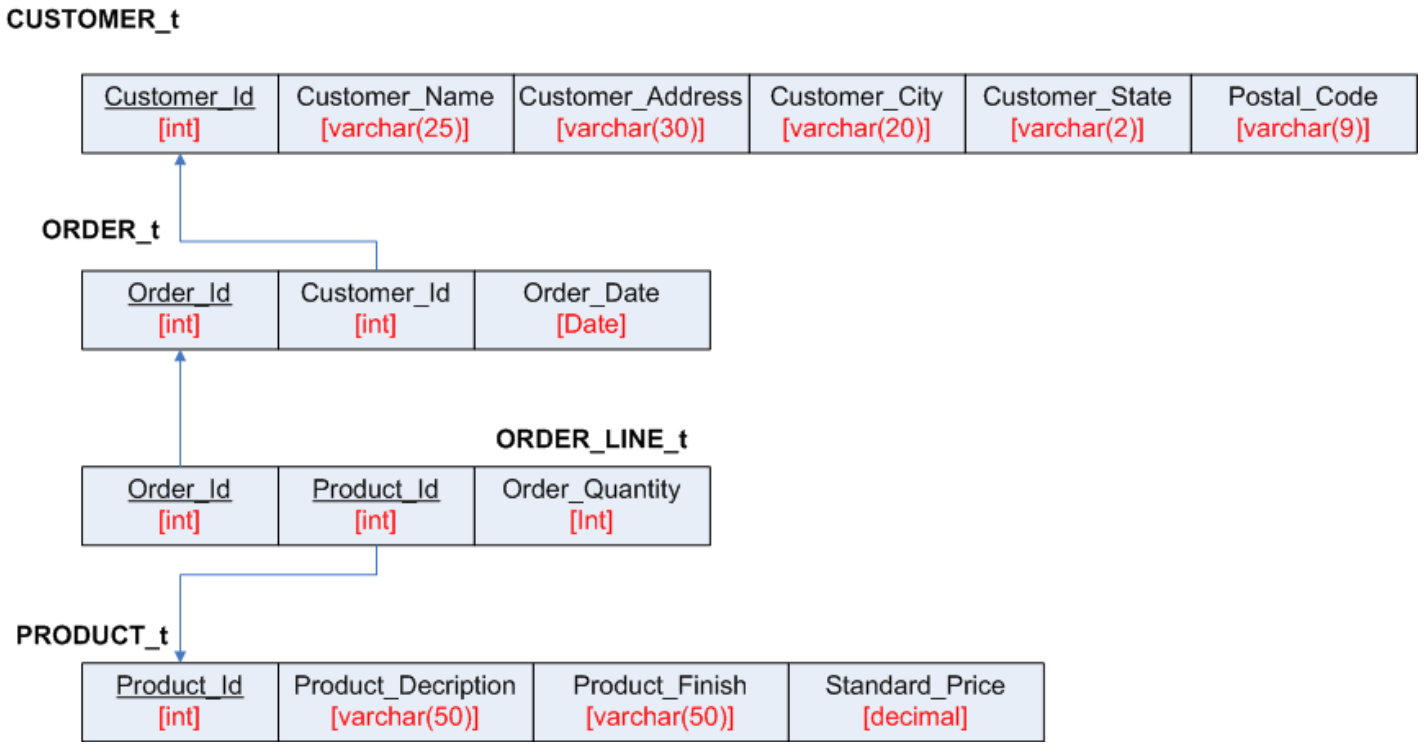
Identifying foreign keys and establishing relationships

```
CREATE TABLE CUSTOMER_T
        (CUSTOMER_ID            NUMBER(11, 0) NOT NULL,
         CUSTOMER_NAME          VARCHAR2(25) NOT NULL,
         CUSTOMER_ADDRESS       VARCHAR2(30),
         CITY                   VARCHAR2(20),
         STATE                  VARCHAR2(2),
         POSTAL_CODE            VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
         (ORDER_ID              NUMBER(11, 0) NOT NULL,
          ORDER_DATE            DATE          DEFAULT SYSDATE,
          CUSTOMER_ID           NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));
```

Primary key of parent table

Foreign key of dependent table

Example


CUSTOMER_t
Customer_Id [int]
Customer_Name [varchar(25)]
Customer_Address [varchar(30)]
Customer_City [varchar(20)]
Customer_State [varchar(2)]
Postal_Code [varchar(9)]
ORDER_t
Order_Id [int]
Customer_Id [int]
Order_Date [Date]
ORDER_LINE_t
Order_Id [int]
Product_Id [int]
Order_Quantity [Int]
PRODUCT_t
Product_Id [int]
Product_Decription [varchar(50)]
Product_Finish [varchar(50)]
Standard_Price [decimal]

Solution

Format 1:

```
CREATE TABLE CUSTOMER_t
(
        Customer_Id             INT     NOT NULL,
        Customer_Name           VARCHAR(25),
        Customer_Address        VARCHAR(30) ,
        Customer_City           VARCHAR(20) ,
        Customer_State          VARCHAR(2) ,
        Postal_Code             VARCHAR(9) ,

    CONSTRAINT CUSTOMER_PK PRIMARY KEY (Customer_Id)
);
```

Format 2:

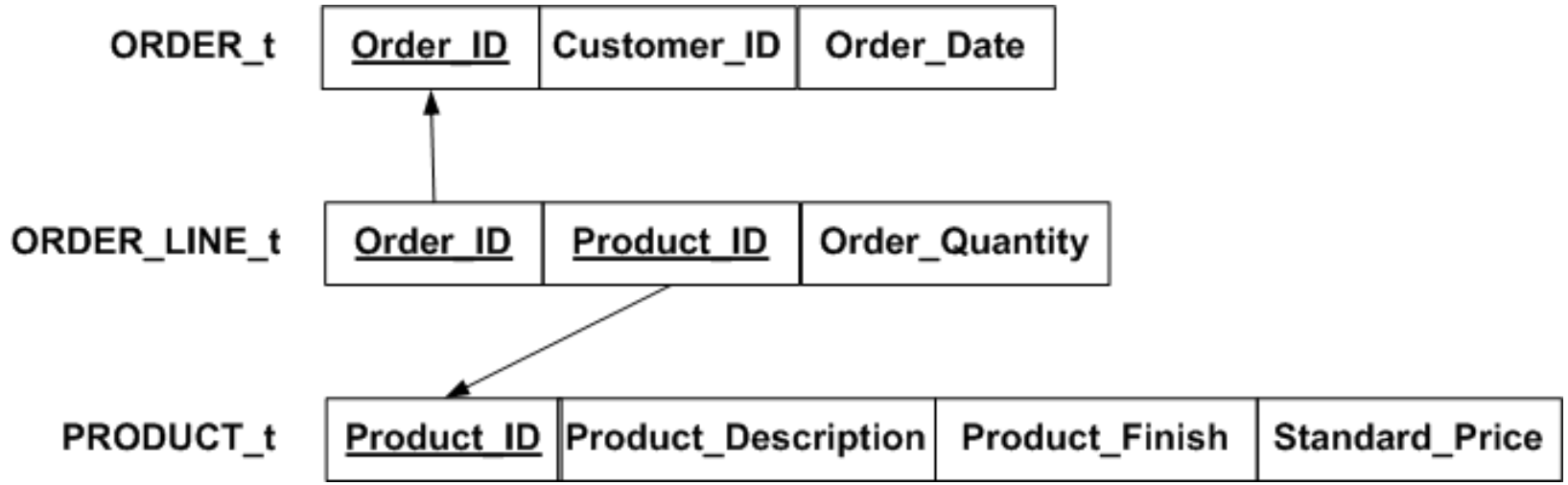
```
CREATE TABLE CUSTOMER_t
(
        Customer_Id             INT     NOT NULL PRIMARY KEY,
        Customer_Name           VARCHAR(25),
        Customer_Address        VARCHAR(30) ,
        Customer_City           VARCHAR(20) ,
        Customer_State          VARCHAR(2) ,
        Postal_Code             VARCHAR(9)
);
```

Note 1: above statement create a table with 6 columns. Customer_Id is the primary key. All the words in red color are reserved KEY WORD.

Note 2: CUSTOMER_PK is the name of one constraint. Within a database, there might be many constraints and they are distinguished from each other by different names. We use CUSTOMER_PK constraint to define the primary key, but it is optional. Format 2 lists an alternative to define the primary key without using CONSTRAINT.

Note 3: The CONSTRAINT constraint_name is optional. In other words,
CONSTRAINT CUSTOMER_PK PRIMARY KEY (Customer_Id) == PRIMARY KEY (Customer_Id)

*WORKSHOP 1 Product key



WORKSHOP 1 Solution

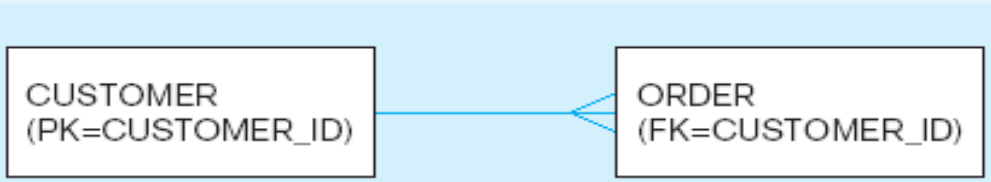
```
CREATE TABLE ORDER_LINE_t
(
        Order_Id            INT     NOT NULL,
        Product_Id          INT     NOT NULL,
        Order_Quantity      INT,

    CONSTRAINT ORDER_LINE_PK PRIMARY KEY (Order_Id, Product_Id),
    CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY (Order_Id)
                REFERENCES ORDER_t(Order_Id),
    CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (Product_Id)
                REFERENCES PRODUCT_t(Product_Id)
);
```

Note: above statement creates a relation called ORDER_LINE_t. It contains 3 columns, as defined in the following figure. The primary key is composite and it contains two attributes Order_Id and Product_id. Meanwhile, both of them are foreign key as well. To define a composite primary key, you can use one constraint, put all the components inside the brackets, and separate them using comma. However, to define multiple foreign keys, you have to use multiple constraints.

Data Integrity Controls

- **Referential integrity**–constraint that ensures that foreign key values of a table must match primary key values of a related table in 1:M relationships
- Restricting:
  - Deletes of primary records
  - Updates of primary records
  - Inserts of dependent records

Figure 7-7 Ensuring data integrity through updates



Restricted Update: A customer ID can only be deleted if it is not found in ORDER table.

```
CREATE TABLE CUSTOMER_T
            (CUSTOMER_ID        INTEGER DEFAULT 'C999'   NOT NULL,
             CUSTOMER_NAME      VARCHAR(40)              NOT NULL,
             . . .
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID),
ON UPDATE RESTRICT);
```

Cascaded Update: Changing a customer ID in the CUSTOMER table will result in that value changing in the ORDER table to match.

. . . ON UPDATE CASCADE);

Set Null Update: When a customer ID is changed, any customer ID in the ORDER table that matches the old customer ID is set to NULL.

. . . ON UPDATE SET NULL);

Set Default Update: When a customer ID is changed, any customer ID in the ORDER tables that matches the old customer ID is set to a predefined default value.

. . . ON UPDATE SET DEFAULT);

Relational integrity is enforced via the primary-key to foreign-key match