INTEGRITY CONSTRAINTS:

INTEGRITY CONSTRAINTS: The relational data model includes several types of constraints, or rules limiting acceptable values and actions, whose purpose is to facilitate maintaining the accuracy and integrity of data in the database.
The three major types of constraints are…

Domain constraints: is the set of values that may be assigned to an attribute

- It usually consists of domain name, meaning, data type, size, and allowable values or ranges.
- Example

TABLE 4-1 Domain Definitions for INVOICE Attributes

Attribute	Domain Name	Description	Domain
CustomerID	Customer IDs	Set of all possible customer IDs	character: size 5
CustomerName	Customer Names	Set of all possible customer names	character: size 25
CustomerAddress	Customer Addresses	Set of all possible customer addresses	character: size 30
CustomerCity	Cities	Set of all possible cities	character: size 20
CustomerState	States	Set of all possible states	character: size 2
CustomerPostalCode	Postal Codes	Set of all possible postal zip codes	character: size 10
OrderID	Order IDs	Set of all possible order IDs	character: size 5
OrderDate	Order Dates	Set of all possible order dates	date: format mm/dd/yy
ProductID	Product IDs	Set of all possible product IDs	character: size 5
ProductDescription	Product Descriptions	Set of all possible product descriptions	character: size 25
ProductFinish	Product Finishes	Set of all possible product finishes	character: size 15
ProductStandardPrice	Unit Prices	Set of all possible unit prices	monetary: 6 digits
ProductLineID	Product Line IDs	Set of all possible product line IDs	integer: 3 digits
OrderedQuantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

Entity integrity: is designed to ensure that every relation has a primary key and that the data values for that primary key are all valid.

- In particular, it guarantees that every primary key attribute is non-null.
  - There are times when other attributes that are not the primary key can be null (absence of value)

Referential integrity: is a rule that maintains consistency among the rows of two relations.

- The rule states that if there is a foreign key in one relation, either each foreign key value must match a primary key value in another relation or the foreign key value must be null.

Creating Relational Tables

In this section, we create table definitions for the four tables shown in Figure 4-5. These definitions are created using the **CREATE TABLE** statements of the SQL data definition language. In practice, these table definitions are actually created during the implementation phase later in the database development process. However, we show these sample tables in this chapter for continuity and especially to illustrate the way the integrity constraints described previously are implemented in SQL.

The SQL table definitions are shown in Figure 4-6. One table is created for each of the four relations shown in the relational schema (Figure 4-5). Each attribute for a table is then defined. Notice that the data type and length for each attribute are taken from the domain definitions (Table 4-1). For example, the attribute CustomerName in the Customer_T table is defined as VARCHAR (variable character) data type with length 25. By specifying **NOT NULL**, each attribute can be constrained from being assigned a null value.

The primary key is specified for each table using the **PRIMARY KEY** clause at the end of each table definition. The OrderLine_T table illustrates how to specify a primary key when that key is a composite attribute. In this example, the primary key of OrderLine_T is the combination of OrderID and ProductID. Each primary key attribute in the four tables is constrained with **NOT NULL**. This enforces the entity integrity constraint described in the previous section. Notice that the **NOT NULL** constraint can also be used with non-primary-key attributes.

Referential integrity constraints are easily defined, based on the graphical schema shown in Figure 4-5. An arrow originates from each foreign key and points to the related primary key in the associated relation. In the SQL table definition, a **FOREIGN KEY REFERENCES** statement corresponds to each of these arrows. Thus, for the table Order_T, the foreign key CustomerID references the primary key of Customer_T, which is also called CustomerID. Although in this case the foreign key and primary key have the same name, this is not required. For example, the foreign key attribute could be named CustNo instead of CustomerID. However, the foreign and primary keys must be from the same domain (i.e., they must have the same data type).

The OrderLine_T table provides an example of a table that has two foreign keys. Foreign keys in this table reference the Order_T and Product_T tables, respectively. Note that these two foreign keys are also the components of the primary key of OrderLine_T. This type of structure is very common as an implementation of a many-to-many relationship.

Note:

- Here is an example of creating a relational table using sql code.

```
CREATE TABLE Customer_T
        (CustomerID                 NUMBER(11,0)     NOT NULL,
        CustomerName                VARCHAR2(25)     NOT NULL,
        CustomerAddress             VARCHAR2(30),
        CustomerCity                VARCHAR2(20),
        CustomerState               CHAR(2),
        CustomerPostalCode          VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
        (OrderID                    NUMBER(11,0)     NOT NULL,
        OrderDate                   DATE DEFAULT SYSDATE,
        CustomerID                  NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID));

CREATE TABLE Product_T
        (ProductID                  NUMBER(11,0)     NOT NULL,
        ProductDescription          VARCHAR2(50),
        ProductFinish               VARCHAR2(20),
        ProductStandardPrice        DECIMAL(6,2),
        ProductLineID               NUMBER(11,0),
CONSTRAINT Product_PK PRIMARY KEY (ProductID));

CREATE TABLE OrderLine_T
        (OrderID                    NUMBER(11,0)     NOT NULL,
        ProductID                   NUMBER(11,0)     NOT NULL,
        OrderedQuantity             NUMBER(11,0),
CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T (OrderID),
CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T (ProductID));
```

Well-Structured Relations: contains minimal redundancy and allows users to insert, modify, and delete the rows in a table without errors or inconsistencies.

- Avoid doing this like this:

EMPLOYEE2

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/2018
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/2018
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/2018
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/2018
110	Chris Lucero	Info Systems	43,000	C++	4/22/2018
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/2018
150	Susan Martin	Marketing	42,000	Java	8/12/2018

- Redundancies in a table that result in errors or inconsistencies are called anomalies. There are three types of anomalies:
    - Insertion anomaly: if a user has to enter data for some attribute that is not necessary like having an employee have to add a course title when it isnt necessary for an employee to have to have taken a course.
    - Deletion anomaly: when you delete a record which results in loss of data that shouldn't have been strictly tied with that record i.e it could have been its own record in another more specific table
        - Suppose that the data for employee number 140 are deleted from the table. This will result in losing the information that this employee com pleted a course (Tax Acc) on 12/8/2018. In fact, it results in losing the information that this course had an offering that completed on that date.
    - Modification anomaly: when you have to update the record of multiple rows because they have the same ID.
    - These are all indications that a table is not well structured