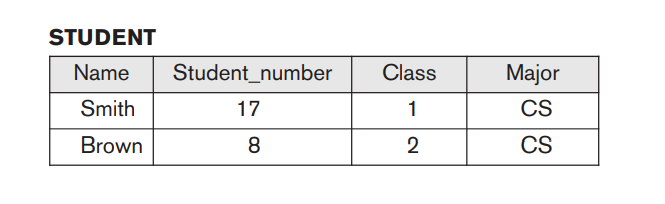
**The Relational Data Model**

The relational model represents the database as a collection of relations. When a relation is thought of as a table of values, each row in the table represents a collection of related data values. The table name and column names are used to help to interpret the meaning of the values in each row.

For example, the first table of Figure 1.2 is called STUDENT because each row represents facts about a particular student entity. The column names—Name, Student\_number, Class, and Major.



In the formal relational model terminology, a row is called a tuple, a column header is called an attribute, and the table is called a relation. The data type describing the types of values that can appear in each column is represented by a domain of possible values.

**Domains, Attributes, Tuples, and Relations**

**Domain:** A domain D is a set of atomic values. By atomic we mean that each value in the domain is indivisible as far as the formal relational model is concerned.

Some examples of domains follow:

■ Usa\_phone\_numbers. The set of ten-digit phone numbers valid in the United States.

■ Local\_phone\_numbers. The set of seven-digit phone numbers valid within a particular area code in the United States.

■ Social\_security\_numbers. The set of valid nine-digit Social Security numbers.

(This is a unique identifier assigned to each person in the United States for employment, tax, and benefits purposes.)

■ Names: The set of character strings that represent names of persons.

■ Grade\_point\_averages. Possible values of computed grade point averages; each must be a real (floating-point) number between 0 and 4.

■ Employee\_ages. Possible ages of employees in a company; each must be an integer value between 15 and 80.

The preceding are called logical definitions of domains. A data type or format is also specified for each domain.

For example, the data type for the domain

Usa\_phone\_numbers can be declared as a character string of the form (ddd)ddd-dddd,

where each d is a numeric (decimal) digit and the first three digits form a valid telephone area code. The data type for Employee\_ages is an integer number between15 and 80.

For Academic\_department\_names, the data type is the set of all character strings that represent valid department names. A domain is thus given a name, data type, and format.

A relation schema R, denoted by R(A1, A2, … , An), is made up of a relation name R and a list of attributes, A1, A2, … , An. Each attribute Ai is the name of a role played by some domain D in the relation schema R. D is called the domain of Ai and is denoted by dom(Ai).

A relation schema is used to describe a relation; R is called the name of this relation. The degree (or arity) of a relation is the number of attributes n of its relation schema.

A relation of degree seven, which stores information about university students,

would contain seven attributes describing each student as follows:

STUDENT(Name, Ssn, Home\_phone, Address, Office\_phone, Age, Gpa)

Using the data type of each attribute, the definition is sometimes written as:

STUDENT(Name: string, Ssn: string, Home\_phone: string, Address: string,Office\_phone: string, Age: integer, Gpa: real)

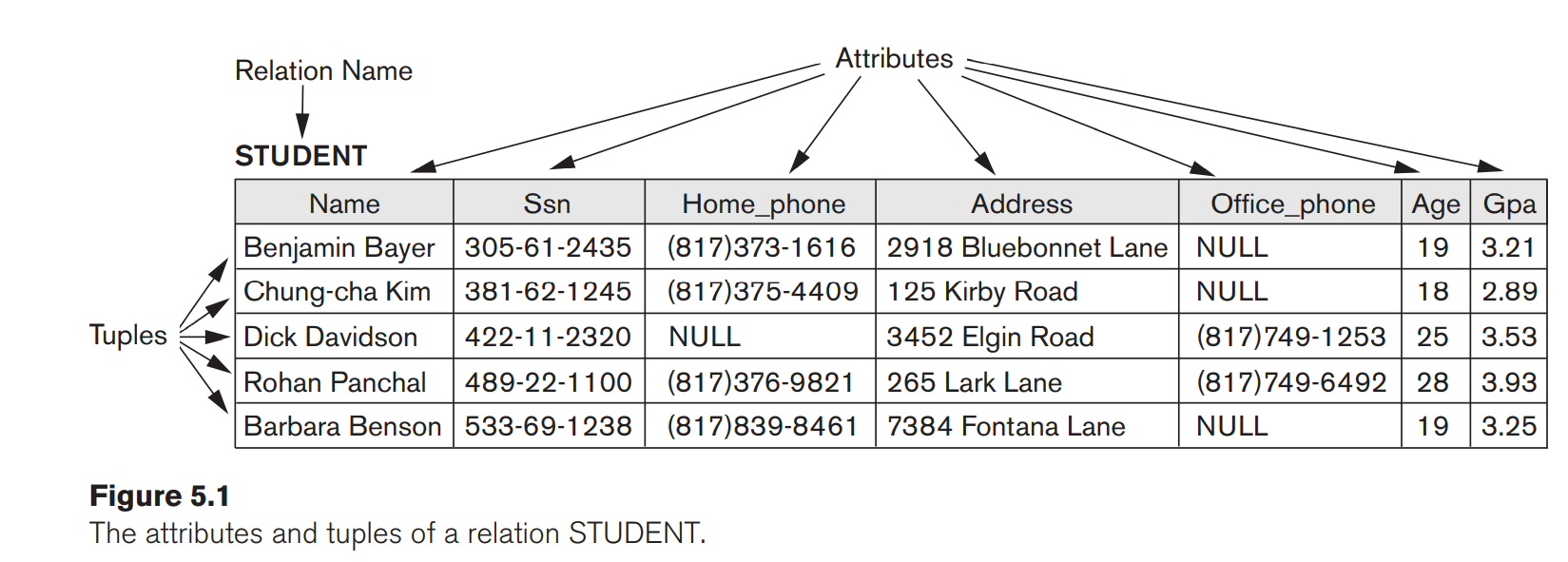
Defined domains for some of the attributes of the STUDENT relation:

dom(Name) = Names;

dom(Ssn) = Social\_security\_numbers;

dom(HomePhone) = USA\_phone\_numbers;

dom(Office\_phone) = USA\_phone\_numbers, and dom(Gpa) = Grade\_point\_averages.



The earlier definition of a relation can be restated more formally using set theory concepts as follows.

A relation (or relation state) r(R) is a mathematical relation of degree n on the domains dom(A1), dom(A2), … , dom(An), which is a subset of theCartesian product (denoted by ×) of the domains that define R:

r(R) ⊆ (dom(A1) × dom(A2) × . . . × (dom(An))

**Relational Model Notation**

A relation schema R of degree n is denoted by R(A1, A2, … , An).

The uppercase letters Q, R, S denote relation names.

The lowercase letters q, r, s denote relation states.

In general, the name of a relation schema such as STUDENT also indicates the current set of tuples in that relation—the current relation state—whereas STUDENT(Name, Ssn, …) refers only to the relation schema.

■ An attribute A can be qualified with the relation name R to which it belongs by using the dot notation R.A—for example, STUDENT.Name or STUDENT.Age. This is because the same name may be used for two attributes in different relations. However, all attribute names in a particular relation must be distinct.

The letters t, u, v denote tuples.

example, consider the tuple t = <’Barbara Benson’, ‘533-69-1238’,‘(817)839-8461’, ‘7384 Fontana Lane’, NULL, 19, 3.25> from the STUDENT relation in Figure 5.1;

we have t[Name] = <‘Barbara Benson’>, and t[Ssn, Gpa, Age] = <‘533-69-1238’,3.25, 19>

**Relational Model Constraints** There are generally many restrictions or constraints on the actual values in a database state. The schema-based constraints include domain constraints, key constraints, constraints on NULLs, entity integrity constraints, and referential integrity constraints.**Domain Constraints**

Domain constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain dom(A). The data types associated with domains typically include standard numeric data types for integers (such as short integer, integer, and long integer) and real numbers (float and double-precision float), Characters, Booleans, fixed-length strings, and variable-length strings are also available, as are date, time, timestamp, and other special data types.

**Constraints on NULL Values**

Constraint on attributes specifies whether NULL values are or are not permitted.

For example, if every STUDENT tuple must have a valid, non-NULL value for the Name attribute, then Name of STUDENT is constrained to be NOT NULL.