Chapter 2
Background

As indicated in the previous chapter, there are two technological bases for distributed database technology: database management and computer networks. In this chapter, we provide an overview of the concepts in these two fields that are more important from the perspective of distributed database technology.

2.1 Overview of Relational DBMS

The aim of this section is to define the terminology and framework used in subsequent chapters, since most of the distributed database technology has been developed using the relational model. In later chapters, when appropriate, we introduce other models. Our focus here is on the language and operators.

2.1.1 Relational Database Concepts

A *database* is a structured collection of data related to some real-life phenomena that we are trying to model. A *relational database* is one where the database structure is in the form of tables. Formally, a relation R defined over n sets $D_1, D_2, \ldots, D_n$ (not necessarily distinct) is a set of *n-tuples* (or simply *tuples*) $\langle d_1, d_2, \ldots, d_n \rangle$ such that $d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n$.

Example 2.1. As an example we use a database that models an engineering company. The entities to be modeled are the *employees* (EMP) and *project*s (PROJ). For each employee, we would like to keep track of the employee number (ENO), name (ENAME), title in the company (TITLE), salary (SAL), identification number of the project(s) the employee is working on (PNO), responsibility within the project (RESP), and duration of the assignment to the project (DUR) in months. Similarly, for each project we would like to store the project number (PNO), the project name (PNAME), and the project budget (BUDGET).

M.T. Özsu and P. Valduriez, *Principles of Distributed Database Systems: Third Edition*,
DOI 10.1007/978-1-4419-8834-8_2, © Springer Science+Business Media, LLC 2011

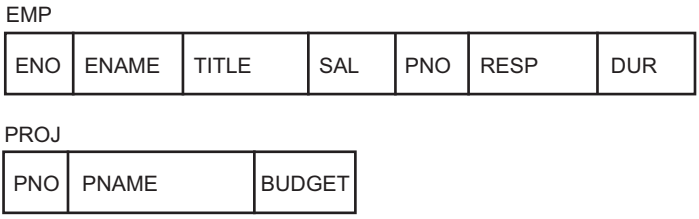

Fig. 2.1 Sample Database Scheme

The *relation schemas* for this database can be defined as follows:

EMP(ENO, ENAME, TITLE, SAL, PNO, RESP, DUR)

PROJ(PNO,PNAME, BUDGET)

In relation scheme EMP, there are seven *attributes*: ENO, ENAME, TITLE, SAL, PNO, RESP, DUR. The values of ENO come from the *domain* of all valid employee numbers, say D_1, the values of ENAME come from the domain of all valid names, say D_2, and so on. Note that each attribute of each relation does not have to come from a distinct domain. Various attributes within a relation or from a number of relations may be defined over the same domain. ♦

The *key* of a relation scheme is the minimum non-empty subset of its attributes such that the values of the attributes comprising the key uniquely identify each tuple of the relation. The attributes that make up key are called *prime* attributes. The superset of a key is usually called a *superkey*. Thus in our example the key of PROJ is PNO, and that of EMP is the set (ENO, PNO). Each relation has at least one key. Sometimes, there may be more than one possibility for the key. In such cases, each alternative is considered a *candidate key*, and one of the candidate keys is chosen as the *primary key*, which we denote by underlining. The number of attributes of a relation defines its *degree*, whereas the number of tuples of the relation defines its *cardinality*.

In tabular form, the example database consists of two tables, as shown in Figure 2.1. The columns of the tables correspond to the attributes of the relations; if there were any information entered as the rows, they would correspond to the tuples. The empty table, showing the structure of the table, corresponds to the *relation schema*; when the table is filled with rows, it corresponds to a *relation instance*. Since the information within a table varies over time, many instances can be generated from one relation scheme. Note that from now on, the term *relation* refers to a relation instance. In Figure 2.2 we depict instances of the two relations that are defined in Figure 2.1.

An attribute value may be undefined. This lack of definition may have various interpretations, the most common being “unknown” or “not applicable”. This special value of the attribute is generally referred to as the *null value*. The representation of a null value must be different from any other domain value, and special care should be given to differentiate it from zero. For example, value “0” for attribute DUR is

EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	40000	P1	Manager	12
E2	M. Smith	Analyst	34000	P1	Analyst	24
E2	M. Smith	Analyst	34000	P2	Analyst	6
E3	A. Lee	Mech. Eng.	27000	P3	Consultant	10
E3	A. Lee	Mech. Eng.	27000	P4	Engineer	48
E4	J. Miller	Programmer	24000	P2	Programmer	18
E5	B. Casey	Syst. Anal.	34000	P2	Manager	24
E6	L. Chu	Elect. Eng.	40000	P4	Manager	48
E7	R. Davis	Mech. Eng.	27000	P3	Engineer	36
E8	J. Jones	Syst. Anal.	34000	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Fig. 2.2 Sample Database Instance

known information (e.g., in the case of a newly hired employee), while value "null" for DUR means unknown. Supporting null values is an important feature necessary to deal with *maybe* queries [Codd, 1979].

2.1.2 Normalization

The aim of normalization is to eliminate various anomalies (or undesirable aspects) of a relation in order to obtain "better" relations. The following four problems might exist in a relation scheme:

1. *Repetition anomaly.* Certain information may be repeated unnecessarily. Consider, for example, the EMP relation in Figure 2.2. The name, title, and salary of an employee are repeated for each project on which this person serves. This is obviously a waste of storage and is contrary to the spirit of databases.

2. *Update anomaly*. As a consequence of the repetition of data, performing updates may be troublesome. For example, if the salary of an employee changes, multiple tuples have to be updated to reflect this change.
3. *Insertion anomaly*. It may not be possible to add new information to the database. For example, when a new employee joins the company, we cannot add personal information (name, title, salary) to the EMP relation unless an appointment to a project is made. This is because the key of EMP includes the attribute PNO, and null values cannot be part of the key.
4. *Deletion anomaly*. This is the converse of the insertion anomaly. If an employee works on only one project, and that project is terminated, it is not possible to delete the project information from the EMP relation. To do so would result in deleting the only tuple about the employee, thereby resulting in the loss of personal information we might want to retain.

Normalization transforms arbitrary relation schemes into ones without these problems. A relation with one or more of the above mentioned anomalies is split into two or more relations of a higher *normal form*. A relation is said to be in a normal form if it satisfies the conditions associated with that normal form. Codd initially defined the *first*, *second*, and *third* normal forms (1NF, 2NF, and 3NF, respectively). Boyce and Codd [Codd, 1974] later defined a modified version of the third normal form, commonly known as the *Boyce-Codd normal form* (BCNF). This was followed by the definition of the *fourth* (4NF) [Fagin, 1977] and *fifth* normal forms (5NF) [Fagin, 1979].

The normal forms are based on certain dependency structures. BCNF and lower normal forms are based on *functional dependencies* (FDs), 4NF is based on *multi-valued dependencies*, and 5NF is based on *projection-join dependencies*. We only introduce functional dependency, since that is the only relevant one for the example we are considering.

Let R be a relation defined over the set of attributes $A = \{A_1, A_2, \ldots, A_n\}$ and let $X \subset A$, $Y \subset A$. If for each value of X in R, there is only one associated Y value, we say that "X *functionally determines* Y" or that "Y is *functionally dependent* on X." Notationally, this is shown as $X \rightarrow Y$. The key of a relation functionally determines the non-key attributes of the same relation.

Example 2.2. For example, in the PROJ relation of Example 2.1 (one can observe these in Figure 2.2 as well), the valid FD is

PNO $\rightarrow$ (PNAME, BUDGET)

In the EMP relation we have

(ENO, PNO) $\rightarrow$ (ENAME,TITLE,SAL,RESP,DUR)

This last FD is not the only FD in EMP, however. If each employee is given unique employee numbers, we can write

ENO → (ENAME, TITLE, SAL)

(ENO, PNO) → (RESP, DUR)

It may also happen that the salary for a given position is fixed, which gives rise to the FD

TITLE → SAL

♦

We do not discuss the normal forms or the normalization algorithms in detail; these can be found in database textbooks. The following example shows the result of normalization on the sample database that we introduced in Example 2.1.

Example 2.3. The following set of relation schemes are normalized into BCNF with respect to the functional dependencies defined over the relations.

EMP(ENO, ENAME, TITLE)

PAY(TITLE, SAL)

PROJ(PNO, PNAME, BUDGET)

ASG(ENO, PNO, RESP, DUR)

The normalized instances of these relations are shown in Figure 2.3. ♦

2.1.3 Relational Data Languages

Data manipulation languages developed for the relational model (commonly called *query languages*) fall into two fundamental groups: *relational algebra* languages and *relational calculus* languages. The difference between them is based on how the user query is formulated. The relational algebra is procedural in that the user is expected to specify, using certain high-level operators, how the result is to be obtained. The relational calculus, on the other hand, is non-procedural; the user only specifies the relationships that should hold in the result. Both of these languages were originally proposed by Codd [1970], who also proved that they were equivalent in terms of expressive power [Codd, 1972].

2.1.3.1 Relational Algebra

Relational algebra consists of a set of operators that operate on relations. Each operator takes one or two relations as operands and produces a result relation, which, in turn, may be an operand to another operator. These operations permit the querying and updating of a relational database.

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Fig. 2.3 Normalized Relations

There are five fundamental relational algebra operators and five others that can be defined in terms of these. The fundamental operators are *selection*, *projection*, *union*, *set difference*, and *Cartesian product*. The first two of these operators are unary operators, and the last three are binary operators. The additional operators that can be defined in terms of these fundamental operators are *intersection*, $\theta - join$, *natural join*, *semijoin* and *division*. In practice, relational algebra is extended with operators for grouping or sorting the results, and for performing arithmetic and aggregate functions. Other operators, such as *outer join* and *transitive closure*, are sometimes used as well to provide additional functionality. We only discuss the more common operators.

The operands of some of the binary relations should be *union compatible*. Two relations R and S are union compatible if and only if they are of the same degree and the i-th attribute of each is defined over the same domain. The second part of the definition holds, obviously, only when the attributes of a relation are identified by their relative positions within the relation and not by their names. If relative ordering of attributes is not important, it is necessary to replace the second part of the definition by the phrase "the corresponding attributes of the two relations should be defined over the same domain." The correspondence is defined rather loosely here.

Many operator definitions refer to "formula", which also appears in relational calculus expressions we discuss later. Thus, let us define precisely, at this point, what we mean by a formula. We define a formula within the context of first-order predicate

calculus (since we use that formalism later), and follow the notation of Gallaire et al. [1984]. First-order predicate calculus is based on a *symbol alphabet* that consists of (1) variables, constants, functions, and predicate symbols; (2) parentheses; (3) the logical connectors $\wedge$ (and), $\vee$ (or), $\neg$ (not), $\rightarrow$ (implication), and $\leftrightarrow$ (equivalence); and (4) quantifiers $\forall$ (for all) and $\exists$ (there exists). A *term* is either a constant or a variable. Recursively, if f is an n-ary function and $t_1, \ldots, t_n$ are terms, $f(t_1, \ldots, t_n)$ is also a term. An *atomic formula* is of the form $P(t_1, \ldots, t_n)$, where P is an n-ary predicate symbol and the t_i's are terms. A *well-formed formula* (*wff*) can be defined recursively as follows: If w_i and w_j are wffs, then (w_i), $\neg(w_i), (w_i) \wedge (w_j), (w_i) \vee (w_j), (w_i) \rightarrow (w_j)$, and $(w_i) \leftrightarrow (w_j)$ are all wffs. Variables in a wff may be *free* or they may be *bound* by one of the two quantifiers.

Selection.

Selection produces a horizontal subset of a given relation. The subset consists of all the tuples that satisfy a formula (condition). The selection from a relation R is

$$\sigma_F(R)$$

where R is the relation and F is a formula.

The formula in the selection operation is called a *selection predicate* and is an atomic formula whose terms are of the form $A\theta c$, where A is an attribute of R and θ is one of the arithmetic comparison operators $<, >, =, \neq, \leq,$ and $\geq$. The terms can be connected by the logical connectors $\wedge, \vee$, and $\neg$. Furthermore, the selection predicate does not contain any quantifiers.

Example 2.4. Consider the relation EMP shown in Figure 2.3. The result of selecting those tuples for electrical engineers is shown in Figure 2.4. ♦

$\sigma_{\text{TITLE="Elect. Eng."}}(\text{EMP})$

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E6	L. Chu	Elect. Eng.

Fig. 2.4 Result of Selection

Projection.

Projection produces a vertical subset of a relation. The result relation contains only those attributes of the original relation over which projection is performed. Thus the degree of the result is less than or equal to the degree of the original relation.

The projection of relation R over attributes A and B is denoted as

$$\Pi_{A,B}(R)$$

Note that the result of a projection might contain tuples that are identical. In that case the duplicate tuples may be deleted from the result relation. It is possible to specify projection with or without duplicate elimination.

Example 2.5. The projection of relation PROJ shown in Figure 2.3 over attributes PNO and BUDGET is depicted in Figure 2.5. ♦

$\Pi_{\text{PNO,BUDGET}}$ (PROJ)

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000

Fig. 2.5 Result of Projection

Union.

The union of two relations R and S (denoted as $R \cup S$) is the set of all tuples that are in R, or in S, or in both. We should note that R and S should be union compatible. As in the case of projection, the duplicate tuples are normally eliminated. Union may be used to insert new tuples into an existing relation, where these tuples form one of the operand relations.

Set Difference.

The set difference of two relations R and S ($R - S$) is the set of all tuples that are in R but not in S. In this case, not only should R and S be union compatible, but the operation is also asymmetric (i.e., $R - S \neq S - R$). This operation allows the

EMP x PAY

ENO	ENAME	EMP.TITLE	PAY.TITLE	SAL
E1	J. Doe	Elect. Eng.	Elect. Eng.	40000
E1	J. Doe	Elect. Eng.	Syst. Anal.	34000
E1	J. Doe	Elect. Eng.	Mech. Eng.	27000
E1	J. Doe	Elect. Eng.	Programmer	24000
E2	M. Smith	Syst. Anal.	Elect. Eng.	40000
E2	M. Smith	Syst. Anal.	Syst. Anal.	34000
E2	M. Smith	Syst. Anal.	Mech. Eng.	27000
E2	M. Smith	Syst. Anal.	Programmer	24000
E3	A. Lee	Mech. Eng.	Elect. Eng.	40000
E3	A. Lee	Mech. Eng.	Syst. Anal.	34000
E3	A. Lee	Mech. Eng.	Mech. Eng.	27000
E3	A. Lee	Mech. Eng.	Programmer	24000
≈	≈	≈	≈	≈
E8	J. Jones	Syst. Anal.	Elect. Eng.	40000
E8	J. Jones	Syst. Anal.	Syst. Anal.	34000
E8	J. Jones	Syst. Anal.	Mech. Eng.	27000
E8	J. Jones	Syst. Anal.	Programmer	24000

Fig. 2.6 Partial Result of Cartesian Product

deletion of tuples from a relation. Together with the union operation, we can perform modification of tuples by deletion followed by insertion.

Cartesian Product.

The Cartesian product of two relations R of degree k_1 and S of degree k_2 is the set of $(k_1 + k_2)$-tuples, where each result tuple is a concatenation of one tuple of R with one tuple of S, for all tuples of R and S. The Cartesian product of R and S is denoted as $R \times S$.

It is possible that the two relations might have attributes with the same name. In this case the attribute names are prefixed with the relation name so as to maintain the uniqueness of the attribute names within a relation.

Example 2.6. Consider relations EMP and PAY in Figure 2.3. EMP $\times$ PAY is shown in Figure 2.6. Note that the attribute TITLE, which is common to both relations, appears twice, prefixed with the relation name. ♦

Intersection.

Intersection of two relations R and S ($R \cap S$) consists of the set of all tuples that are in both R and S. In terms of the basic operators, it can be specified as follows:

$$R \cap S = R - (R - S)$$

θ-Join.

Join is a derivative of Cartesian product. There are various forms of join; the primary classification is between *inner join* and *outer join*. We first discuss inner join and its variants and then describe outer join.

The most general type of inner join is the θ-join. The θ-join of two relations R and S is denoted as

$$R \bowtie_F S$$

where F is a formula specifying the *join predicate*. A join predicate is specified similar to a selection predicate, except that the terms are of the form $R.A\theta S.B$, where A and B are attributes of R and S, respectively.

The join of two relations is equivalent to performing a selection, using the join predicate as the selection formula, over the Cartesian product of the two operand relations. Thus

$$R \bowtie_F S = \sigma_F(R \times S)$$

In the equivalence above, we should note that if F involves attributes of the two relations that are common to both of them, a projection is necessary to make sure that those attributes do not appear twice in the result.

Example 2.7. Let us consider that the EMP relation in Figure 2.3 and add two more tuples as depicted in Figure 2.7(a). Then Figure 2.7(b) shows the θ-join of relations EMP and ASG over the join predicate EMP.ENO=ASG.ENO.

The same result could have been obtained as

$$\text{EMP} \bowtie_{\text{EMP.ENO=ASG.ENO}} \text{ASG} = \Pi_{\text{ENO, ENAME, TITLE, SAL}}(\sigma_{\text{EMP.ENO =PAY.ENO}}(\text{EMP} \times \text{ASG}))$$

Notice that the result does not have tuples E9 and E10 since these employees have not yet been assigned to a project. Furthermore, the information about some employees (e.g., E2 and E3) who have been assigned to multiple projects appear more than once in the result. ♦

This example demonstrates a special case of θ-join which is called the *equi-join*. This is a case where the formula F only contains equality (=) as the arithmetic operator. It should be noted, however, that an equi-join does not have to be specified over a common attribute as the example above might suggest.

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.
E9	A. Hsu	Programmer
E10	T. Wong	Syst. Anal.

(a)

EMP $\bowtie_{\text{EMP.ENO=ASG.ENO}}$ ASG

ENO	ENAME	TITLE	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	P1	Manager	12
E2	M. Smith	Syst. Anal.	P1	Analyst	12
E2	M. Smith	Syst. Anal.	P2	Analyst	12
E3	A. Lee	Mech. Eng.	P3	Consultant	12
E3	A. Lee	Mech. Eng.	P4	Engineer	12
E4	J. Miller	Programmer	P2	Programmer	12
E5	J. Miller	Syst. Anal.	P2	Manager	12
E6	L. Chu	Elect. Eng.	P4	Manager	12
E7	R. Davis	Mech. Eng.	P3	Engineer	12
E8	J. Jones	Syst. Anal.	P3	Manager	12

(b)

Fig. 2.7 The Result of Join

A natural join is an equi-join of two relations over a specified attribute, more specifically, over attributes with the same domain. There is a difference, however, in that usually the attributes over which the natural join is performed appear only once in the result. A natural join is denoted as the join without the formula

$$R \bowtie_A S$$

where A is the attribute common to both R and S. We should note here that the natural join attribute may have different names in the two relations; what is required is that they come from the same domain. In this case the join is denoted as

$$R_A \bowtie_B S$$

where B is the corresponding join attribute of S.

Example 2.8. The join of EMP and ASG in Example 2.7 is actually a natural join. Here is another example – Figure 2.8 shows the natural join of relations EMP and PAY in Figure 2.3 over the attribute TITLE.

♦

Inner join requires the joined tuples from the two operand relations to satisfy the join predicate. In contrast, outer join does not have this requirement – tuples exist in the result relation regardless. Outer join can be of three types: left outer join ($\overset{\frown}{\bowtie}$), right outer join ($\overset{\frown}{\bowtie}$) and full outer join ($\overset{\frown}{\bowtie}$). In the left outer join, the tuples from the left operand relation are always in the result, in the case of right outer join, the tuples from the right operand are always in the result, and in the case of full outer relation, tuples from both relations are always in the result. Outer join is useful in those cases where we wish to include information from one or both relations even if the do not satisfy the join predicate.

EMP ⋈ TITLE PAY

ENO	ENAME	TITLE	SAL
E1	J. Doe	Elect. Eng.	40000
E2	M. Smith	Analyst	34000
E3	A. Lee	Mech. Eng.	27000
E4	J. Miller	Programmer	24000
E5	B. Casey	Syst. Anal.	34000
E6	L. Chu	Elect. Eng.	40000
E7	R. Davis	Mech. Eng.	27000
E8	J. Jones	Syst. Anal.	34000

Fig. 2.8 The Result of Natural Join

Example 2.9. Consider the left outer join of EMP (as revised in Example 2.7) and ASG over attribute ENO (i.e., EMP $⟕_{ENO}$ ASG). The result is given in Figure 2.9. Notice that the information about two employees, E9 and E10 are included in the result even thought they have not yet been assigned to a project with "Null" values for the attributes from the ASG relation. ♦

EMP⟕ ENO ASG

ENO	ENAME	TITLE	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	P1	Manager	12
E2	M. Smith	Syst. Anal.	P1	Analyst	12
E2	M. Smith	Syst. Anal.	P2	Analyst	12
E3	A. Lee	Mech. Eng.	P3	Consultant	12
E3	A. Lee	Mech. Eng.	P4	Engineer	12
E4	J. Miller	Programmer	P2	Programmer	12
E5	J. Miller	Syst. Anal.	P2	Manager	12
E6	L. Chu	Elect. Eng.	P4	Manager	12
E7	R. Davis	Mech. Eng.	P3	Engineer	12
E8	J. Jones	Syst. Anal.	P3	Manager	12
E9	A. Hsu	Programmer	Null	Null	Null
E10	T. Wong	Syst. Anal.	Null	Null	Null

Fig. 2.9 The Result of Left Outer Join

Semijoin.

The semijoin of relation R, defined over the set of attributes A, by relation S, defined over the set of attributes B, is the subset of the tuples of R that participate in the join of R with S. It is denoted as $R \ltimes_F S$ (where F is a predicate as defined before) and can be obtained as follows:

$$\begin{aligned} R \ltimes_F S &= \Pi_A(R \bowtie_F S) = \Pi_A(R) \bowtie_F \Pi_{A \cap B}(S) \\ &= R \bowtie_F \Pi_{A \cap B}(S) \end{aligned}$$

The advantage of semijoin is that it decreases the number of tuples that need to be handled to form the join. In centralized database systems, this is important because it usually results in a decreased number of secondary storage accesses by making better use of the memory. It is even more important in distributed databases since it usually reduces the amount of data that needs to be transmitted between sites in order to evaluate a query. We talk about this in more detail in Chapters 3 and 8. At this point note that the operation is asymmetric (i.e., $R \ltimes_F S \neq S \ltimes_F R$).

Example 2.10. To demonstrate the difference between join and semijoin, let us consider the semijoin of EMP with PAY over the predicate EMP.TITLE = PAY.TITLE, that is,

$$\text{EMP} \ltimes_{\text{EMP.TITLE = PAY.TITLE}} \text{PAY}$$

The result of the operation is shown in Figure 2.10. We encourage readers to compare Figures 2.7 and 2.10 to see the difference between the join and the semijoin operations. Note that the resultant relation does not have the PAY attribute and is therefore smaller. ♦

EMP $\ltimes_{\text{EMP.TITLE=PAY.TITLE}}$ PAY

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Analyst
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

Fig. 2.10 The Result of Semijoin

Division.

The division of relation R of degree r with relation S of degree s (where $r > s$ and $s \neq 0$) is the set of $(r-s)$-tuples t such that for all s-tuples u in S, the tuple tu is in R. The division operation is denoted as $R \div S$ and can be specified in terms of the fundamental operators as follows:

$$R \div S = \Pi_{\bar{A}}(R) - \Pi_{\bar{A}}((\Pi_{\bar{A}}(R) \times S) - R)$$

where $\bar{A}$ is the set of attributes of R that are not in S [i.e., the $(r-s)$-tuples].

Example 2.11. Assume that we have a modified version of the ASG relation (call it ASG′) depicted in Figure 2.11a and defined as follows:

$$\text{ASG}' = \Pi_{\text{ENO,PNO}}\ (\text{ASG}) \bowtie_{\text{PNO}} \text{PROJ}$$

If one wants to find the employee numbers of those employees who are assigned to all the projects that have a budget greater than \$200,000, it is necessary to divide ASG′ with a restricted version of PROJ, called PROJ′ (see Figure 2.11b). The result of division (ASG′÷ PROJ′) is shown in Figure 2.11c.

The keyword in the query above is "*all*." This rules out the possibility of doing a selection on ASG′ to find the necessary tuples, since that would only give those which correspond to employees working on *some* project with a budget greater than \$200,000, not those who work on all projects. Note that the result contains only the tuple ⟨E3⟩ since the tuples ⟨E3, P3, CAD/CAM, 250000⟩ and ⟨E3, P4, Maintenance, 310000⟩ both exist in ASG′. On the other hand, for example, ⟨E7⟩ is not in the result, since even though the tuple ⟨E7, P3, CAD/CAM, 250000⟩ is in ASG′, the tuple ⟨E7, P4, Maintenance, 310000⟩ is not. ♦

Since all operations take relations as input and produce relations as outputs, we can nest operations using a parenthesized notation and represent relational algebra programs. The parentheses indicate the order of execution. The following are a few examples that demonstrate the issue.

Example 2.12. Consider the relations of Figure 2.3. The retrieval query

"Find the names of employees working on the CAD/CAM project"

can be answered by the relational algebra program

$$\Pi_{\text{ENAME}}(((\sigma_{\text{PNAME = ``CAD/CAM''}}\ \text{PROJ}) \bowtie_{\text{PNO}} \text{ASG}) \bowtie_{\text{ENO}} \text{EMP})$$

The order of execution is: the selection on PROJ, followed by the join with ASG, followed by the join with EMP, and finally the project on ENAME.

An equivalent program where the size of the intermediate relations is smaller is

$$\Pi_{\text{ENAME}}\ (\text{EMP} \ltimes_{\text{ENO}} (\Pi_{\text{ENO}} (\text{ASG} \ltimes_{\text{PNO}} (\sigma_{\text{PNAME= ``CAD/CAM''}}\ \text{PROJ}))))$$

♦

ASG'

ENO	PNO	PNAME	BUDGET
E1	P1	Instrumentation	150000
E2	P1	Instrumentation	150000
E2	P2	Database Develop.	135000
E3	P3	CAD/CAM	250000
E3	P4	Maintenance	310000
E4	P2	Database Develop.	135000
E5	P2	Database Develop.	135000
E6	P4	Maintenance	310000
E7	P3	CAD/CAM	250000
E8	P3	CAD/CAM	250000

(a)

PROJ'

PNO	PNAME	BUDGET
P3	CAD/CAM	250000
P4	Maintenance	310000

(b)

(ASG' ÷ PROJ')

ENO
E3

(c)

Fig. 2.11 The Result of Division

Example 2.13. The update query

"Replace the salary of programmers by $25,000"

can be computed by

$$(\text{PAY} - (\sigma_{\text{TITLE} = \text{“Programmer”}} \text{PAY})) \cup (\langle \text{Programmer}, 25000 \rangle)$$

♦

2.1.3.2 Relational Calculus

In relational calculus-based languages, instead of specifying *how* to obtain the result, one specifies *what* the result is by stating the relationship that is supposed to hold for the result. Relational calculus languages fall into two groups: *tuple relational calculus* and *domain relational calculus*. The difference between the two is in terms

of the primitive variable used in specifying the queries. We briefly review these two types of languages.

Relational calculus languages have a solid theoretical foundation since they are based on first-order predicate logic as we discussed before. Semantics is given to formulas by interpreting them as assertions on the database. A relational database can be viewed as a collection of tuples or a collection of domains. Tuple relational calculus interprets a variable in a formula as a tuple of a relation, whereas domain relational calculus interprets a variable as the value of a domain.

Tuple relational calculus.

The primitive variable used in tuple relational calculus is a *tuple variable* which specifies a tuple of a relation. In other words, it ranges over the tuples of a relation. Tuple calculus is the original relational calculus developed by Codd [1970].

In tuple relational calculus queries are specified as $\{t|F(t)\}$, where t is a tuple variable and F is a well-formed formula. The atomic formulas are of two forms:

1. *Tuple-variable membership expressions*. If t is a tuple variable ranging over the tuples of relation R (predicate symbol), the expression "tuple t belongs to relation R" is an atomic formula, which is usually specified as $R.t$ or $R(t)$.
2. *Conditions*. These can be defined as follows:

   (a) $s[A]\theta t[B]$, where s and t are tuple variables and A and B are components of s and t, respectively. θ is one of the arithmetic comparison operators $<$, $>$, $=$, $\neq$, $\leq$, and $\geq$. This condition specifies that component A of s stands in relation θ to the B component of t: for example, $s[\text{SAL}] > t[\text{SAL}]$.

   (b) $s[A]\theta c$, where s, A, and θ are as defined above and c is a constant. For example, $s[\text{ENAME}] = \text{"Smith"}$.

Note that A is defined as a component of the tuple variable s. Since the range of s is a relation instance, say S, it is obvious that component A of s corresponds to attribute A of relation S. The same thing is obviously true for B.

There are many languages that are based on relational tuple calculus, the most popular ones being SQL[1] [Date, 1987] and QUEL [Stonebraker et al., 1976]. SQL is now an international standard (actually, the only one) with various versions released: SQL1 was released in 1986, modifications to SQL1 were included in the 1989 version, SQL2 was issued in 1992, and SQL3, with object-oriented language extensions, was released in 1999.

[1] Sometimes SQL is cited as lying somewhere between relational algebra and relational calculus. Its originators called it a "mapping language." However, it follows the tuple calculus definition quite closely; hence we classify it as such.

SQL provides a uniform approach to data manipulation (retrieval, update), data definition (schema manipulation), and control (authorization, integrity, etc.). We limit ourselves to the expression, in SQL, of the queries in Examples 2.14 and 2.15.

Example 2.14. The query from Example 2.12,

"Find the names of employees working on the CAD/CAM project"

can be expressed as follows:

```
SELECT  EMP.ENAME
FROM    EMP,ASG,PROJ
WHERE   EMP.ENO = ASG.ENO
AND     ASG.PNO = PROJ.PNO
AND     PROJ.PNAME = "CAD/CAM"
```

♦

Note that a retrieval query generates a new relation similar to the relational algebra operations.

Example 2.15. The update query of Example 2.13,

"Replace the salary of programmers by $25,000"

is expressed as

```
UPDATE  PAY
SET     SAL = 25000
WHERE   PAY.TITLE = "Programmer"
```

♦

Domain relational calculus.

The domain relational calculus was first proposed by Lacroix and Pirotte [1977]. The fundamental difference between a tuple relational language and a domain relational language is the use of a *domain variable* in the latter. A domain variable ranges over the values in a domain and specifies a component of a tuple. In other words, the range of a domain variable consists of the domains over which the relation is defined. The wffs are formulated accordingly. The queries are specified in the following form:

$$x_1, x_2, \ldots, x_n | F(x_1, x_2, \ldots, x_n)$$

where F is a wff in which $x_1, \ldots, x_n$ are the free variables.

The success of domain relational calculus languages is due mainly to QBE [Zloof, 1977], which is a visual application of domain calculus. QBE, designed only for interactive use from a visual terminal, is user friendly. The basic concept is an *example*: the user formulates queries by providing a possible example of the answer. Typing relation names triggers the printing, on screen, of their schemes. Then, by supplying keywords into the columns (domains), the user specifies the query. For instance, the attributes of the project relation are given by P, which stands for "Print."

EMP	ENO	ENAME	TITLE
	E2	P.	

ASG	ENO	PNO	RESP	DUR
	E2	P3		

PROJ	PNO	PNAME	BUDGET
	P3	CAD/CAM	

Fig. 2.12 Retrieval Query in QBE

By default, all queries are retrieval. An update query requires the specification of U under the name of the updated relation or in the updated column. The retrieval query corresponding to Example 2.12 is given in Figure 2.12 and the update query of Example 2.13 is given in Figure 2.13. To distinguish examples from constants, examples are underlined.

PAY	TITLE	SAL
	Programmer	U.25000

Fig. 2.13 Update Query in QBE

2.2 Review of Computer Networks

In this section we discuss computer networking concepts relevant to distributed database systems. We omit most of the details of the technological and technical issues in favor of discussing the main concepts.

We define a *computer network* as an *interconnected collection of autonomous computers that are capable of exchanging information among themselves* (Figure 2.14). The keywords in this definition are *interconnected* and *autonomous*. We want the computers to be autonomous so that each computer can execute programs on its own. We also want the computers to be interconnected so that they are capable of exchanging information. Computers on a network are referred to as *nodes*, *hosts*, *end systems*, or *sites*. Note that sometimes the terms *host* and *end system* are used to refer

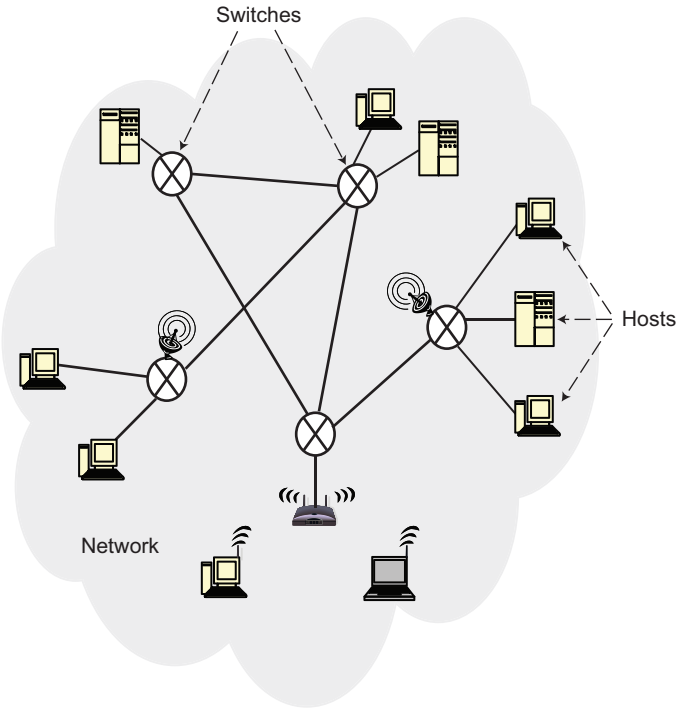

Fig. 2.14 A Computer Network

simply to the equipment, whereas *site* is reserved for the equipment as well as the software that runs on it. Similarly, *node* is generally used as a generic reference to the computers or to the switches in a network. They form one of the fundamental hardware components of a network. The other fundamental component is special purpose devices and links that form the communication path that interconnects the nodes. As depicted in Figure 2.14, the hosts are connected to the network through switches (represented as circles with an X in them)[2], which are special-purpose equipment that *route* messages through the network. Some of the hosts may be connected to the switches directly (using fiber optic, coaxial cable or copper wire) and some via wireless base stations. The switches are connected to each other by communication links that may be fiber optics, coaxial cable, satellite links, microwave connections, etc.

The most widely used computer network these days is the Internet. It is hard to define the Internet since the term is used to mean different things, but perhaps the best definition is that it is a network of networks (Figure 2.15). Each of these

[2] Note that the terms "switch" and "router" are sometimes used interchangeably (even within the same text). However, other times they are used to mean slightly different things: switch refers to the devices inside a network whereas router refers to one that is at the edge of a network connecting it to the backbone. We use them interchangeably as in Figures 2.14 and 2.15.

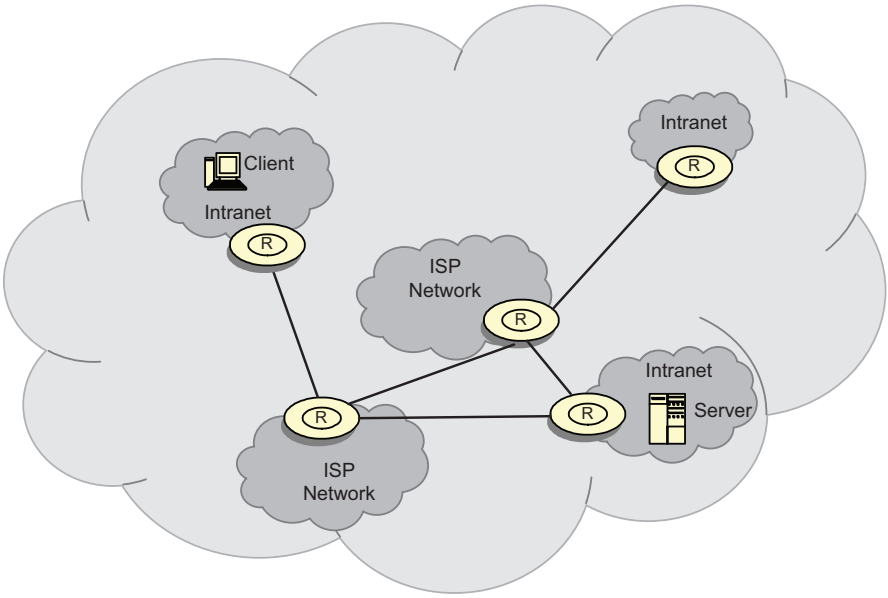

Fig. 2.15 Internet

networks is referred to as an *intranet* to highlight the fact that they are "internal" to an organization. An intranet, then, consists of a set of links and routers (shown as "R" in Figure 2.15) administered by a single administrative entity or by its delegates. For instance, the routers and links at a university constitute a single administrative domain. Such domains may be located within a single geographical area (such as the university network mentioned above), or, as in the case of large enterprises or Internet Service Provider (ISP) networks, span multiple geographical areas. Each intranet is connected to some others by means of links provisioned from ISPs. These links are typically high-speed, long-distance duplex data transmission media (we will define these terms shortly), such as a fiber-optic cable, or a satellite link. These links make up what is called the Internet backbone. Each intranet has a router interface that connects it to the backbone, as shown in Figure 2.15. Thus, each link connects an intranet router to an ISP's router. ISP's routers are connected by similar links to routers of other ISPs. This allows servers and clients within an intranet to communicate with servers and clients in other intranets.

2.2.1 Types of Networks

There are various criteria by which computer networks can be classified. One criterion is the geographic distribution (also called *scale* [Tanenbaum, 2003]), a second

criterion is the *interconnection structure* of nodes (also called *topology*), and the third is the mode of transmission.

2.2.1.1 Scale

In terms of geographic distribution, networks are classified as wide area networks, metropolitan area networks and local area networks. The distinctions among these are somewhat blurred, but in the following, we give some general guidelines that identify each of these networks. The primary distinction among them are probably in terms of propagation delay, administrative control, and the protocols that are used in managing them.

A wide area network (WAN) is one where the link distance between any two nodes is greater than approximately 20 kilometers (km) and can go as large as thousands of kilometers. Use of switches allow the aggregation of communication over wider areas such as this. Owing to the distances that need to be traveled, long delays are involved in wide area data transmission. For example, via satellite, there is a minimum delay of half a second for data to be transmitted from the source to the destination and acknowledged. This is because the speed with which signals can be transmitted is limited to the speed of light, and the distances that need to be spanned are great (about 31,000 km from an earth station to a satellite).

WANs are typically characterized by the heterogeneity of the transmission media, the computers, and the user community involved. Early WANs had a limited capacity of less than a few megabits-per-second (Mbps). However, most of the current ones are broadband WANs that provide capacities of 150 Mbps and above. These individual channels are aggregated into the backbone links; the current backbone links are commonly OC48 at 2.4 Gbps or OC192 at 10Gbps. These networks can carry multiple data streams with varying characteristics (e.g., data as well as audio/video streams), the possibility of negotiating for a level of quality of service (QoS) and reserving network resources sufficient to fulfill this level of QoS.

Local area networks (LANs) are typically limited in geographic scope (usually less than 2 km). They provide higher capacity communication over inexpensive transmission media. The capacities are typically in the range of 10-1000 Mbps per connection. Higher capacity and shorter distances between hosts result in very short delays. Furthermore, the better controlled environments in which the communication links are laid out (within buildings, for example) reduce the noise and interference, and the heterogeneity among the computers that are connected is easier to manage, and a common transmission medium is used.

Metropolitan area networks (MANs) are in between LANs and WANs in scale and cover a city or a portion of it. The distances between nodes is typically on the order of 10 km.

2.2.1.2 Topology

As the name indicates, interconnection structure or topology refers to the way nodes on a network are interconnected. The network in Figure 2.14 is what is called an *irregular* network, where the interconnections between nodes do not follow any pattern. It is possible to find a node that is connected to only one other node, as well as nodes that have connections to a number of nodes. Internet is a typical irregular network.

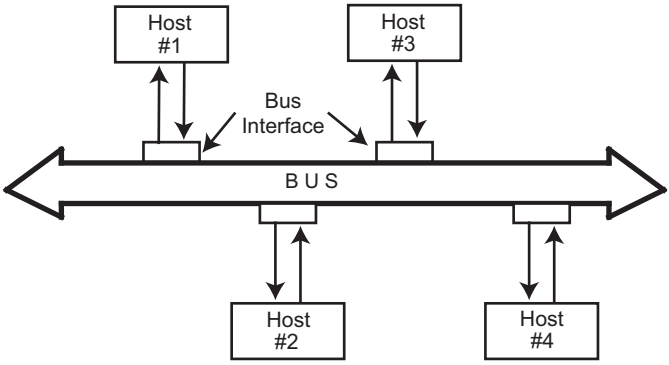

Fig. 2.16 Bus Network

Another popular topology is the bus, where all the computers are connected to a common channel (Figure 2.16). This type of network is primarily used in LANs. The link control is typically performed using *carrier sense medium access with collision detection* (CSMA/CD) protocol. The CSMA/CD bus control mechanism can best be described as a "listen before and while you transmit" scheme. The fundamental point is that each host listens continuously to what occurs on the bus. When a message transmission is detected, the host checks if the message is addressed to it, and takes the appropriate action. If it wants to transmit, it waits until it detects no more activity on the bus and then places its message on the network and continues to listen to bus activity. If it detects another transmission while it is transmitting a message itself, then there has been a "collision." In such a case, and when the collision is detected, the transmitting hosts abort the transmission, each waits a random amount of time, and then each retransmits the message. The basic CSMA/CD scheme is used in the Ethernet local area network[3].

Other common alternatives are star, ring, bus, and mesh networks.

[3] In most current implementations of Ethernet, multiple busses are linked via one or more switches (called *switched hubs*) for expanded coverage and to better control the load on each bus segment. In these systems, individual computers can directly be connected to the switch as well. These are known as switched Ethernet.

- *Star* networks connect all the hosts to a central node that coordinates the transmission on the network. Thus if two hosts want to communicate, they have to go through the central node. Since there is a separate link between the central node and each of the others, there is a negotiation between the hosts and the central node when they wish to communicate.
- *Ring* networks interconnect the hosts in the form of a loop. This type of network was originally proposed for LANs, but their use in these networks has nearly stopped. They are now primarily used in MANs (e.g., SONET rings). In their current incarnation, data transmission around the ring is usually bidirectional (original rings were unidirectional), with each station (actually the interface to which each station is connected) serving as an active repeater that receives a message, checks the address, copies the message if it is addressed to that station, and retransmits it.
  Control of communication in ring type networks is generally controlled by means of a *control token*. In the simplest type of token ring networks, a token, which has one bit pattern to indicate that the network is free and a different bit pattern to indicate that it is in use, is circulated around the network. Any site wanting to transmit a message waits for the token. When it arrives, the site checks the token's bit pattern to see if the network is free or in use. If it is free, the site changes the bit pattern to indicate that the network is in use and then places the messages on the ring. The message circulates around the ring and returns to the sender which changes the bit pattern to free and sends the token to the next computer down the line.
- *Complete* (or *mesh*) interconnection is one where each node is interconnected to every other node. Such an interconnection structure obviously provides more reliability and the possibility of better performance than that of the structures noted previously. However, it is also the costliest. For example, a complete connection of 10,000 computers would require approximately $(10,000)^2$ links.[4]

2.2.2 Communication Schemes

In terms of the physical communication schemes employed, networks can be either *point-to-point* (also called *unicast*) networks, or *broadcast* (sometimes also called *multi-point*) networks.

In point-to-point networks, there are one or more (direct or indirect) links between each pair of nodes. The communication is always between two nodes and the receiver and sender are identified by their addresses that are included in the message header. Data transmission from the sender to the receiver follows one of the possibly many links between them, some of which may involve visiting other intermediate nodes. An intermediate node checks the destination address in the message header and if it is not addressed to it, passes it along to the next intermediate node. This is the

[4] The general form of the equation is $n(n-1)/2$, where n is the number of nodes on the network.

process of *switching* or *routing*. The selection of the links via which messages are sent is determined by usually elaborate routing algorithms that are beyond our scope. We discuss the details of switching in Section 2.2.3.

The fundamental transmission media for point-to-point networks are twisted pair, coaxial or fiber optic cables. Each of these media have different capacities: twisted pair 300 bps to 10 Mbps, coaxial up to 200 Mbps, and fiber optic 10 Gbps and even higher.

In broadcast networks, there is a common communication channel that is utilized by all the nodes in the network. Messages are transmitted over this common channel and received by all the nodes. Each node checks the receiver address and if the message is not addressed to it, ignores it.

A special case of broadcasting is *multicasting* where the message is sent to a subset of the nodes in the network. The receiver address is somehow encoded to indicate which nodes are the recipients.

Broadcast networks are generally radio or satellite-based. In case of satellite transmission, each site beams its transmission to a satellite which then beams it back at a different frequency. Every site on the network listens to the receiving frequency and has to disregard the message if it is not addressed to that site. A network that uses this technique is HughesNet™.

Microwave transmission is another mode of data communication and it can be over satellite or terrestrial. Terrestrial microwave links used to form a major portion of most countries' telephone networks although many of these have since been converted to fiber optic. In addition to the public carriers, some companies make use of private terrestrial microwave links. In fact, major metropolitan cities face the problem of microwave interference among privately owned and public carrier links. A very early example that is usually identified as having pioneered the use of satellite microwave transmission is ALOHA [Abramson, 1973].

Satellite and microwave networks are examples of wireless networks. These types of wireless networks are commonly referred to as *wireless broadband* networks. Another type of wireless network is one that is based on *cellular* networks. A cellular network control station is responsible for a geographic area called a *cell* and coordinates the communication from mobile hosts in their cell. These control stations may be linked to a "wireline" backbone network and thereby provide access from/to mobile hosts to other mobile hosts or stationary hosts on the wireline network.

A third type of wireless network with which most of us may be more familiar are *wireless LANs* (commonly referred to as Wi-LAN or WiLan). In this case a number of "base stations" are connected to a wireline network and serve as connection points for mobile hosts (similar to control stations in cellular networks). These networks can provide bandwidth of up to 54 Mbps.

A final word on broadcasting topologies is that they have the advantage that it is easier to check for errors and to send messages to more than one site than to do so in point-to-point topologies. On the other hand, since everybody listens in, broadcast networks are not as secure as point-to-point networks.

2.2.3 Data Communication Concepts

What we refer to as data communication is the set of technologies that enable two hosts to communicate. We are not going to be too detailed in this discussion, since, at the distributed DBMS level, we can assume that the technology exists to move bits between hosts. We, instead, focus on a few important issues that are relevant to understanding delay and routing concepts.

As indicated earlier hosts are connected by *links*, each of which can carry one or more *channels*. Link is a physical entity whereas channel is a logical one. Communication links can carry signals either in digital form or in analog form. Telephone lines, for example, can carry data in analog form between the home and the central office – the rest of the telephone network is now digital and even the home-to-central office link is becoming digital with voice-over-IP (VoIP) technology. Each communication channel has a *capacity*, which can be defined as the amount of information that can be transmitted over the channel in a given time unit. This capacity is commonly referred to as the *bandwidth* of the channel. In analog transmission channels, the bandwidth is defined as the difference (in hertz) between the lowest and highest frequencies that can be transmitted over the channel per second. In digital links, *bandwidth* refers (less formally and with abuse of terminology) to the number of bits that can be transmitted per second (bps).

With respect to delays in getting the user's work done, the bandwidth of a transmission channel is a significant factor, but it is not necessarily the only ones. The other factor in the transmission time is the software employed. There are usually overhead costs involved in data transmission due to the redundancies within the message itself, necessary for error detection and correction. Furthermore, the network software adds headers and trailers to any message, for example, to specify the destination or to check for errors in the entire message. All of these activities contribute to delays in transmitting data. The actual rate at which data are transmitted across the network is known as the *data transfer rate* and this rate is usually less than the actual bandwidth of the transmission channel. The software issues, that generally are referred as *network protocols*, are discussed in the next section.

In computer-to-computer communication, data are usually transmitted in *packets*, as we mentioned earlier. Usually, upper limits on frame sizes are established for each network and each contains data as well as some control information, such as the destination and source addresses, block error check codes, and so on (Figure 2.17). If a message that is to be sent from a source node to a destination node cannot fit one frame, it is split over a number of frames. This is be discussed further in Section 2.2.4.

There are various possible forms of switching/routing that can occur in point-to-point networks. It is possible to establish a connection such that a dedicated channel exists between the sender and the receiver. This is called *circuit switching* and is commonly used in traditional telephone connections. When a subscriber dials the number of another subscriber, a circuit is established between the two phones by means of various switches. The circuit is maintained during the period of conversation and is broken when one side hangs up. Similar setup is possible in computer networks.

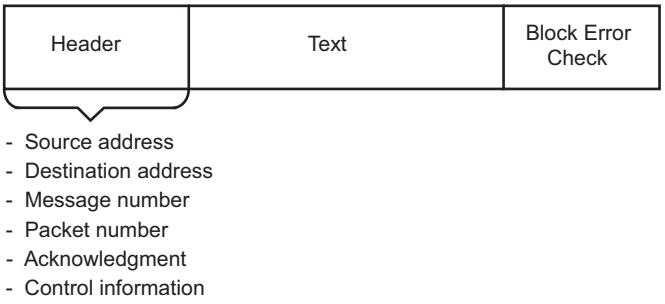

Fig. 2.17 Typical Frame Format

Another form of switching used in computer communication is *packet switching*, where a message is broken up into packets and each packet transmitted individually. In our discussion of the TCP/IP protocol earlier, we referred to messages being transmitted; in fact the TCP protocol (or any other transport layer protocol) takes each application package and breaks it up into fixed sized packets. Therefore, each application message may be sent to the destination as multiple packets.

Packets for the same message may travel independently of each other and may, in fact, take different routes. The result of routing packets along possibly different links in the network is that they may arrive at the destination out-of-order. Thus the transport layer software at the destination site should be able to sort them into their original order to reconstruct the message. Consequently, it is the individual packages that are routed through the network, which may result in packets reaching the destination at different times and even out of order. The transport layer protocol at the destination is responsible for collating and ordering the packets and generating the application message properly.

The advantages of packet switching are many. First, packet-switching networks provide higher link utilization since each link is not dedicated to a pair of communicating equipment and can be shared by many. This is especially useful in computer communication due to its bursty nature – there is a burst of transmission and then some break before another burst of transmission starts. The link can be used for other transmission when it is idle. Another reason is that packetizing may permit the parallel transmission of data. There is usually no requirement that various packets belonging to the same message travel the same route through the network. In such a case, they may be sent in parallel via different routes to improve the total data transmission time. As mentioned above, the result of routing frames this way is that their in-order delivery cannot be guaranteed.

On the other hand, circuit switching provides a dedicated channel between the receiver and the sender. If there is a sizable amount of data to be transmitted between the two or if the channel sharing in packet switched networks introduces too much delay or delay variance, or packet loss (which are important in multimedia applications), then the dedicated channel facilitates this significantly. Therefore, schemes similar to circuit switching (i.e., reservation-based schemes) have gained favor in

the broadband networks that support applications such as multimedia with very high data transmission loads.

2.2.4 Communication Protocols

Establishing a physical connection between two hosts is not sufficient for them to communicate. Error-free, reliable and efficient communication between hosts requires the implementation of elaborate software systems that are generally called *protocols*. Network protocols are "layered" in that network functionality is divided into layers, each layer performing a well-defined function relying on the services provided by the layer below it and providing a service to the layer above. A protocol defines the services that are performed at one layer. The resulting layered protocol set is referred to as a *protocol stack* or *protocol suite*.

There are different protocol stacks for different types of networks; however, for communication over the Internet, the standard one is what is referred to as TCP/IP that stands for "Transport Control Protocol/Internet Protocol". We focus primarily on TCP/IP in this section as well as some of the common LAN protocols.

Before we get into the specifics of the TCP/IP protocol stack, let us first discuss how a message from a process on host C in Figure 2.15 is transmitted to a process on server S, assuming both hosts implement the TCP/IP protocol. The process is depicted in Figure 2.18.

The appropriate application layer protocol takes the message from the process on host C and creates an application layer message by adding some application layer header information (oblique hatched part in Figure 2.18) details of which are not important for us. The application message is handed over to the TCP protocol, which repeats the process by adding its own header information. TCP header includes the necessary information to facilitate the provision of TCP services we discuss shortly. The Internet layer takes the TCP message that is generated and forms an Internet message as we also discuss below. This message is now physically transmitted from host C to its router using the protocol of its own network, then through a series of routers to the router of the network that contains server S, where the process is reversed until the original message is recovered and handed over to the appropriate process on S. The TCP protocols at hosts C and S communicate to ensure the end-to-end guarantees that we discussed.

2.2.4.1 TCP/IP Protocol Stack

What is referred to as TCP/IP is in fact a family of protocols, commonly referred to as the *protocol stack*. It consists of two sets of protocols, one set at the *transport layer* and the other at the *network (Internet) layer* (Figure 2.19).

The transport layer defines the types of services that the network provides to applications. The protocols at this layer address issues such as data loss (can the

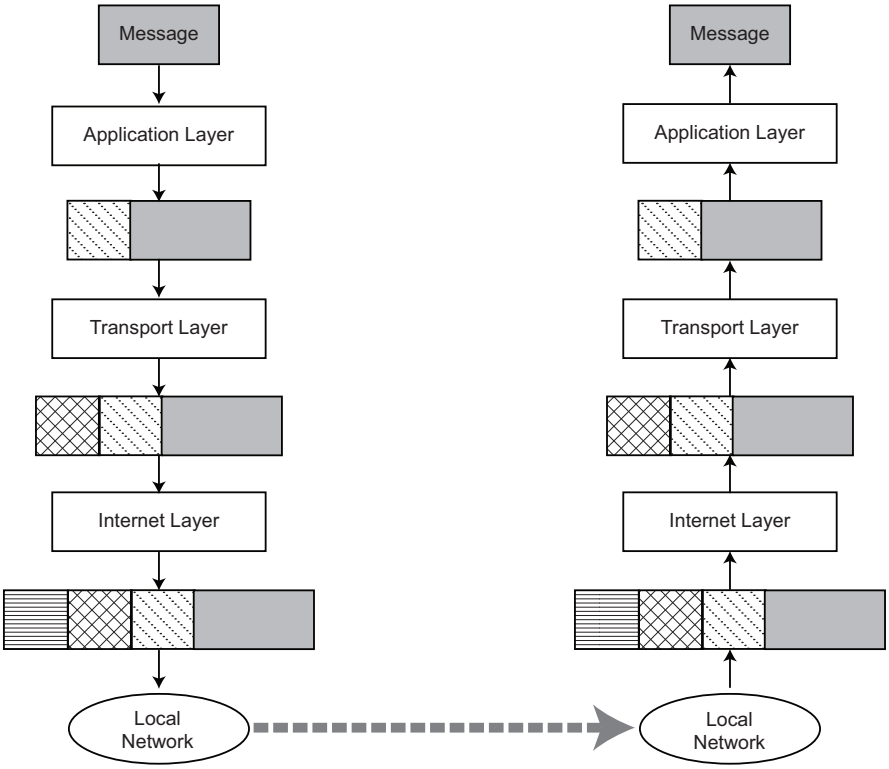

Fig. 2.18 Message Transmission using TCP/IP

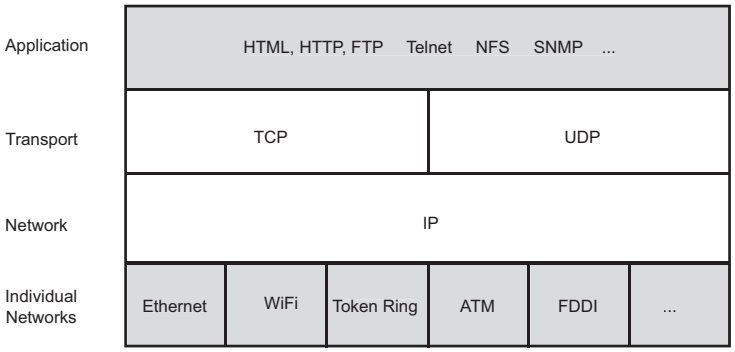

Fig. 2.19 TCP/IP Protocol

application tolerate losing some of the data during transmission?), bandwidth (some applications have minimum bandwidth requirements while others can be more elastic in their requirements), and timing (what type of delay can the applications tolerate?). For example, a file transfer application can not tolerate any data loss, can be flexible in its bandwidth use (it will work whether the connection is high capacity or low capacity, although the performance may differ), and it does not have strict timing requirements (although we may not like a file transfer to take a few days, it would still work). In contrast, a real-time audio/video transmission application can tolerate a limited amount of data loss (this may cause some jitter and other problems, but the communication will still be "understandable"), has minimum bandwidth requirement (5-128 Kbps for audio and 5 Kbps-20 Mbps for video), and is time sensitive (audio and video data need to be synchronized).

To deal with these varying requirements (at least with some of them), at the transport layer, two protocols are provided: TCP and UDP. TCP is connection-oriented, meaning that prior setup is required between the sender and the receiver before actual message transmission can start; it provides reliable transmission between the sender and the receiver by ensuring that the messages are received correctly at the receiver (referred to as "end-to-end reliability"); ensures flow control so that the sender does not overwhelm the receiver if the receiver process is not able to keep up with the incoming messages, and ensures congestion control so that the sender is throttled when network is overloaded. Note that TCP does not address the timing and minimum bandwidth guarantees, leaving these to the application layer.

UDP, on the other hand, is a connectionless service that does not provide the reliability, flow control and congestion control guarantees that TCP provides. Nor does it establish a connection between the sender and receiver beforehand. Thus, each message is transmitted hoping that it will get to the destination, but no end-to-end guarantees are provided. Thus, UDP has significantly lower overhead than TCP, and is preferred by applications that would prefer to deal with these requirements themselves, rather than having the network protocol handle them.

The network layer implements the Internet Protocol (IP) that provides the facility to "package" a message in a standard Internet message format for transmission across the network. Each Internet message can be up to 64KB long and consists of a header that contains, among other things, the IP addresses of the sender and the receiver machines (the numbers such as 129.97.79.58 that you may have seen attached to your own machines), and the message body itself. The message format of each network that makes up the Internet can be different, but each of these messages are encoded into an Internet message by the Internet Protocol before they are transmitted[5].

The importance of TCP/IP is the following. Each of the intranets that are part of the Internet can use its own preferred protocol, so the computers on that network implement that particular protocol (e.g., the token ring mechanism and the CSMA/CS technique described above are examples of these types of protocols). However, if they are to connect to the Internet, they need to be able to communicate using TCP/IP, which are implemented on top of these specific network protocols (Figure 2.19).

[5] Today, many of the Intranets also use TCP/IP, in which case IP encapsulation may not be necessary.

2.2.4.2 Other Protocol Layers

Let us now briefly consider the other two layers depicted in Figure 2.19. Although these are not part of the TCP/IP protocol stack, they are necessary to be able to build distributed applications. These make up the top and the bottom layers of the protocol stack.

The Application Protocol layer provides the specifications that distributed applications have to follow. For example, if one is building a Web application, then the documents that will be posted on the Web have to be written according to the HTML protocol (note that HTML is not a networking protocol, but a document encoding protocol) and the communication between the client browser and the Web server has to follow the HTTP protocol. Similar protocols are defined at this layer for other applications as indicated in the figure.

The bottom layer represents the specific network that may be used. Each of those networks have their own message formats and protocols and they provide the mechanisms for data transmission within those networks.

The standardization for LANs is spearheaded by the Institute of Electrical and Electronics Engineers (IEEE), specifically their Committee No. 802; hence the standard that has been developed is known as the IEEE 802 Standard. The three layers of the IEEE 802 local area network standard are the physical layer, the medium access control layer, and the logical link control layer.

The physical layer deals with physical data transmission issues such as signaling. Medium access control layer defines protocols that control who can have access to the transmission medium and when. Logical link control layer implements protocols that ensure reliable packet transmission between two adjacent computers (not end-to-end). In most LANs, the TCP and IP layer protocols are implemented on top of these three layers, enabling each computer to be able to directly communicate on the Internet.

To enable it to cover a variety of LAN architectures, the 802 local area network standard is actually a number of standards rather than a single one. Originally, it was specified to support three mechanisms at the medium access control level: the CSMA/CD mechanism, token ring, and token access mechanism for bus networks.

2.3 Bibliographic Notes

This chapter covered the basic issues related to relational database systems and computer networks. These concepts are discussed in much greater detail in a number of excellent textbooks. Related to database technology, we can name [Ramakrishnan and Gehrke, 2003; Elmasri and Navathe, 2011; Silberschatz et al., 2002; Garcia-Molina et al., 2002; Kifer et al., 2006], and [Date, 2004]. For computer networks one can refer to [Tanenbaum, 2003; Kurose and Ross, 2010; Leon-Garcia and Widjaja, 2004; Comer, 2009]. More focused discussion of data communication issues can be found in [Stallings, 2011].

http://www.springer.com/978-1-4419-8833-1

Principles of Distributed Database Systems
Özsu, M.T.; Valduriez, P.
2011, XX, 846 p., Hardcover
ISBN: 978-1-4419-8833-1