views in relational systems. In Section 6.4 we examine views in more detail and show how they can be created and used within SQL.

3.4.1 Terminology

The relations we have been dealing with so far in this chapter are known as base relations.

Base relation A named relation corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database.

We can define views in terms of base relations:

View The dynamic result of one or more relational operations operating on the base relations to produce another relation. A view is a *virtual relation* that does not necessarily exist in the database but can be produced upon request by a particular user, at the time of request.

A view is a relation that appears to the user to exist, can be manipulated as if it were a base relation, but does not necessarily exist in storage in the sense that the base relations do (although its definition is stored in the system catalog). The contents of a view are defined as a query on one or more base relations. Any operations on the view are automatically translated into operations on the relations from which it is derived. Views are **dynamic**, meaning that changes made to the base relations that affect the view are immediately reflected in the view. When users make permitted changes to the view, these changes are made to the underlying relations. In this section, we describe the purpose of views and briefly examine restrictions that apply to updates made through views. However, we defer treatment of how views are defined and processed until Section 6.4.

3.4.2 Purpose of Views

The view mechanism is desirable for several reasons:

- It provides a powerful and flexible security mechanism by hiding parts of the database from certain users. Users are not aware of the existence of any attributes or tuples that are missing from the view.
- It permits users to access data in a way that is customized to their needs, so that the same data can be seen by different users in different ways, at the same time.
- It can simplify complex operations on the base relations. For example, if a view is defined as a combination (join) of two relations (see Section 4.1), users may now perform more simple operations on the view, which will be translated by the DBMS into equivalent operations on the join.

A view should be designed to support the external model that the user finds familiar. For example:

- A user might need Branch tuples that contain the names of managers as well as the other attributes already in Branch. This view is created by combining the Branch relation with a restricted form of the Staff relation where the staff position is 'Manager'.
- Some members of staff should see Staff tuples without the salary attribute.
- Attributes may be renamed or the order of attributes changed. For example, the user accustomed to calling the branchNo attribute of branches by the full name Branch Number may see that column heading.
- Some members of staff should see only property records for those properties that they manage.

Although all these examples demonstrate that a view provides *logical data independence* (see Section 2.1.5), views allow a more significant type of logical data independence that supports the reorganization of the conceptual schema. For example, if a new attribute is added to a relation, existing users can be unaware of its existence if their views are defined to exclude it. If an existing relation is rearranged or split up, a view may be defined so that users can continue to see their original views. We will see an example of this in Section 6.4.7 when we discuss the advantages and disadvantages of views in more detail.

Updating Views 3.4.3

All updates to a base relation should be immediately reflected in all views that reference that base relation. Similarly, if a view is updated, then the underlying base relation should reflect the change. However, there are restrictions on the types of modification that can be made through views. We summarize below the conditions under which most systems determine whether an update is allowed through a view:

- Updates are allowed through a view defined using a simple query involving a single base relation and containing either the primary key or a candidate key of the base relation.
- Updates are not allowed through views involving multiple base relations.
- Updates are not allowed through views involving aggregation or grouping operations.

Classes of views have been defined that are **theoretically not updatable**, **theoretically updatable**, and **partially updatable**. A survey on updating relational views can be found in Furtado and Casanova (1985).

Chapter Summary

- The Relational Database Management System (RDBMS) has become the dominant data-processing software in use today, with estimated new licence sales of between US$6 billion and US$10 billion per year (US$25 billion with tools sales included). This software represents the second generation of DBMSs and is based on the relational data model proposed by E. F. Codd.
- A mathematical **relation** is a subset of the Cartesian product of two or more sets. In database terms, a relation is any subset of the Cartesian product of the domains of the attributes. A relation is normally written as a set of *n*-tuples, in which each element is chosen from the appropriate domain.
- Relations are physically represented as **tables**, with the rows corresponding to individual tuples and the columns to attributes.
- The structure of the relation, with domain specifications and other constraints, is part of the **intension** of the database, while the relation with all its tuples written out represents an **instance** or **extension** of the database.
- Properties of database relations are: each cell contains exactly one atomic value, attribute names are distinct, attribute values come from the same domain, attribute order is immaterial, tuple order is immaterial, and there are no duplicate tuples.
- The **degree** of a relation is the number of attributes, while the **cardinality** is the number of tuples. A **unary** relation has one attribute, a **binary** relation has two, a **ternary** relation has three, and an ***n*-ary** relation has *n* attributes.
- A **superkey** is an attribute, or set of attributes, that identifies tuples of a relation uniquely, while a **candidate key** is a minimal superkey. A **primary key** is the candidate key chosen for use in identification of tuples. A relation must always have a primary key. A **foreign key** is an attribute, or set of attributes, within one relation that is the candidate key of another relation.
- A **null** represents a value for an attribute that is unknown at the present time or is not applicable for this tuple.
- **Entity integrity** is a constraint that states that in a base relation no attribute of a primary key can be null. **Referential integrity** states that foreign key values must match a candidate key value of some tuple in the home relation or be wholly null. Apart from relational integrity, integrity constraints include, required data, domain, and multiplicity constraints; other integrity constraints are called **general constraints**.
- A **view** in the relational model is a **virtual** or **derived relation** that is dynamically created from the underlying base relation(s) when required. Views provide security and allow the designer to customize a user's model. Not all views are updatable.