Database Management Systems
Chapter 5: Physical Database Design

UNIVERSITY OF
MARYLAND

ROBERT H. SMITH
SCHOOL OF BUSINESS

DR. ADAM LEE

Objectives

- Define terms.
- Describe the physical database design process.
- Choose storage formats for attributes.
- Select appropriate file organizations.
- Describe three types of file organization
- Describe indexes and their appropriate use.
- Translate a database model into efficient structures.
- Know when and how to use denormalization.

Physical Database Design

- **Purpose** – translate the logical description of data into the technical specifications for storing and retrieving data.
- **Goal** – create a design for storing data that will provide adequate performance and ensure database integrity, security, and recoverability.

Physical Database Design

Inputs

- Normalized relations
- Volume estimates
- Attribute definitions
- Response time expectations
- Data security needs
- Backup/recovery needs
- Integrity expectations
- DBMS technology used

Leads to

Decisions

- Attribute data types
- Physical record descriptions (doesn't always match logical design)
- File organizations
- Indexes and database architectures
- Query optimization

UNIVERSITY OF MARYLAND

Figure 5-1: Composite Usage Map (Pine Valley Furniture Company)

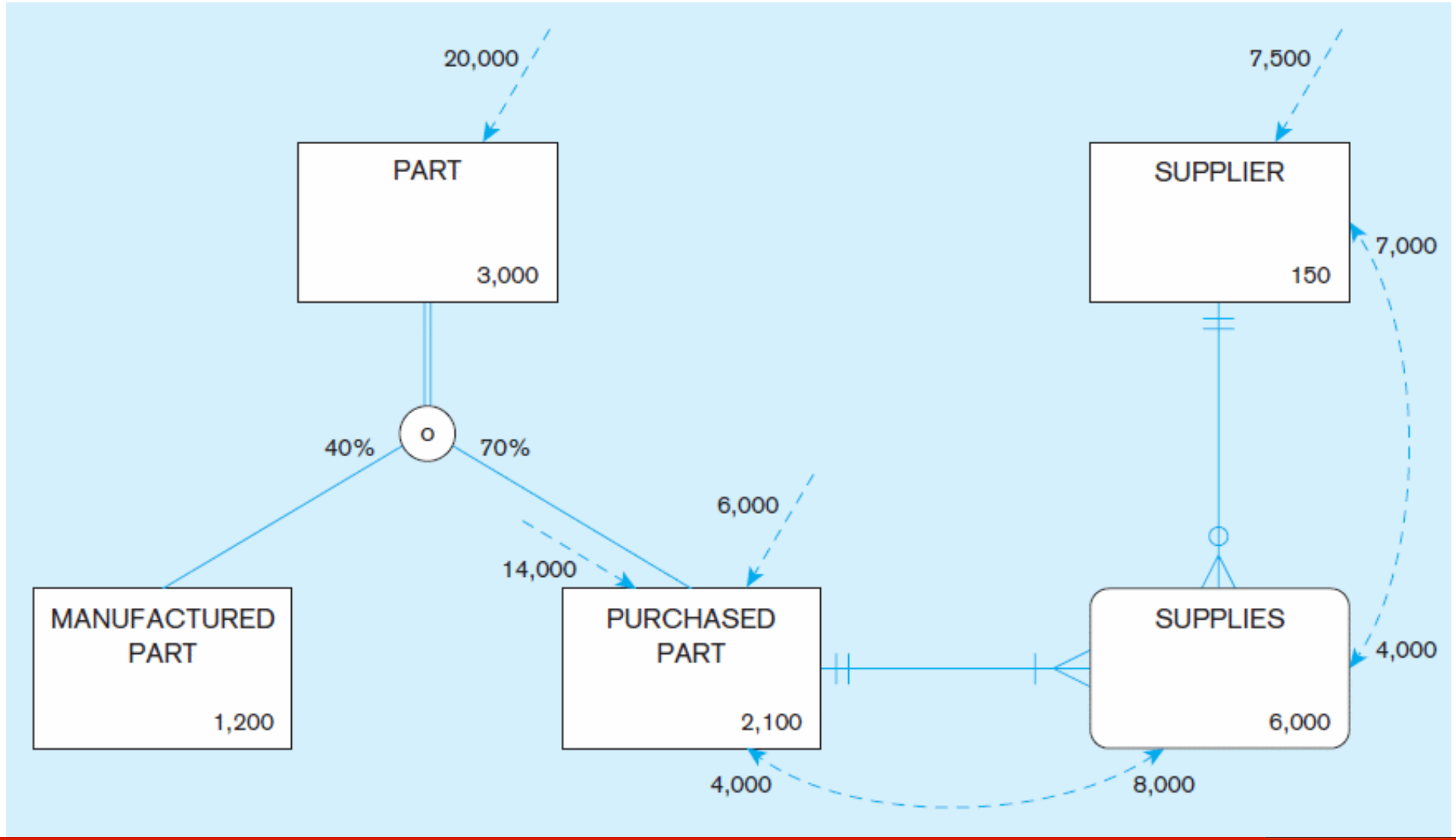


Figure 5-1: Composite Usage Map (Pine Valley Furniture Company)

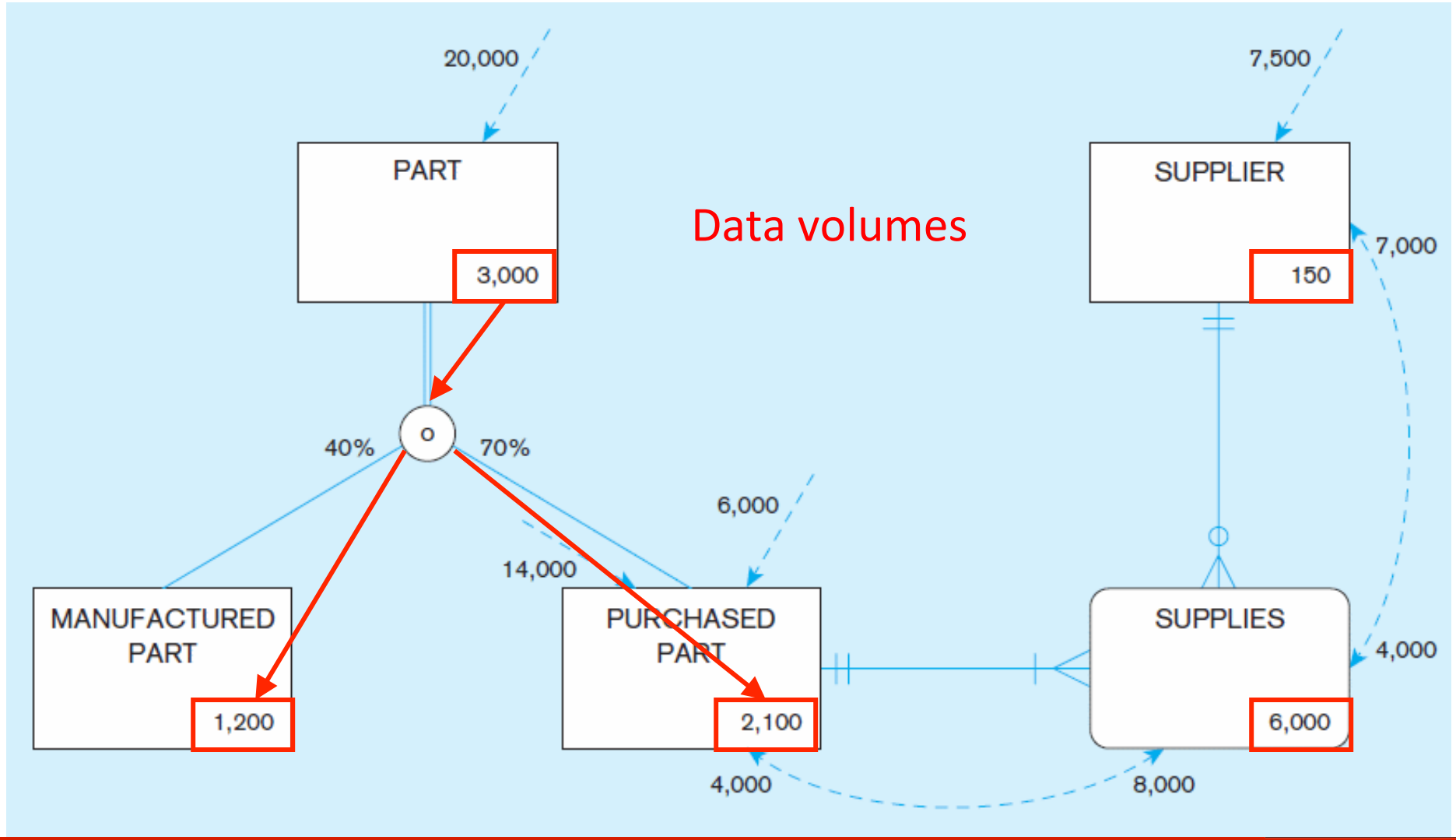


Figure 5-1: Composite Usage Map (Pine Valley Furniture Company)

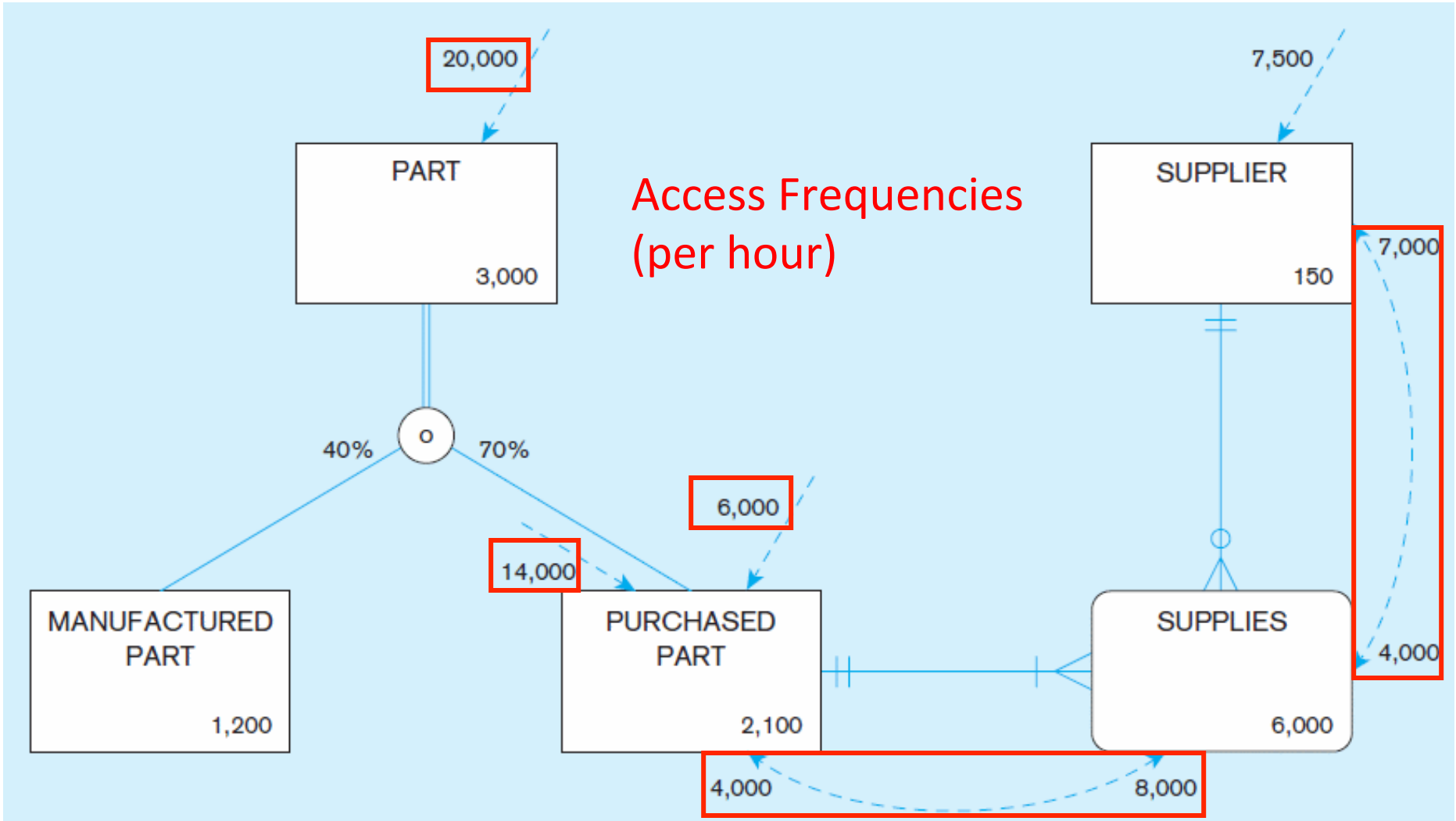


Figure 5-1: Composite Usage Map (Pine Valley Furniture Company)

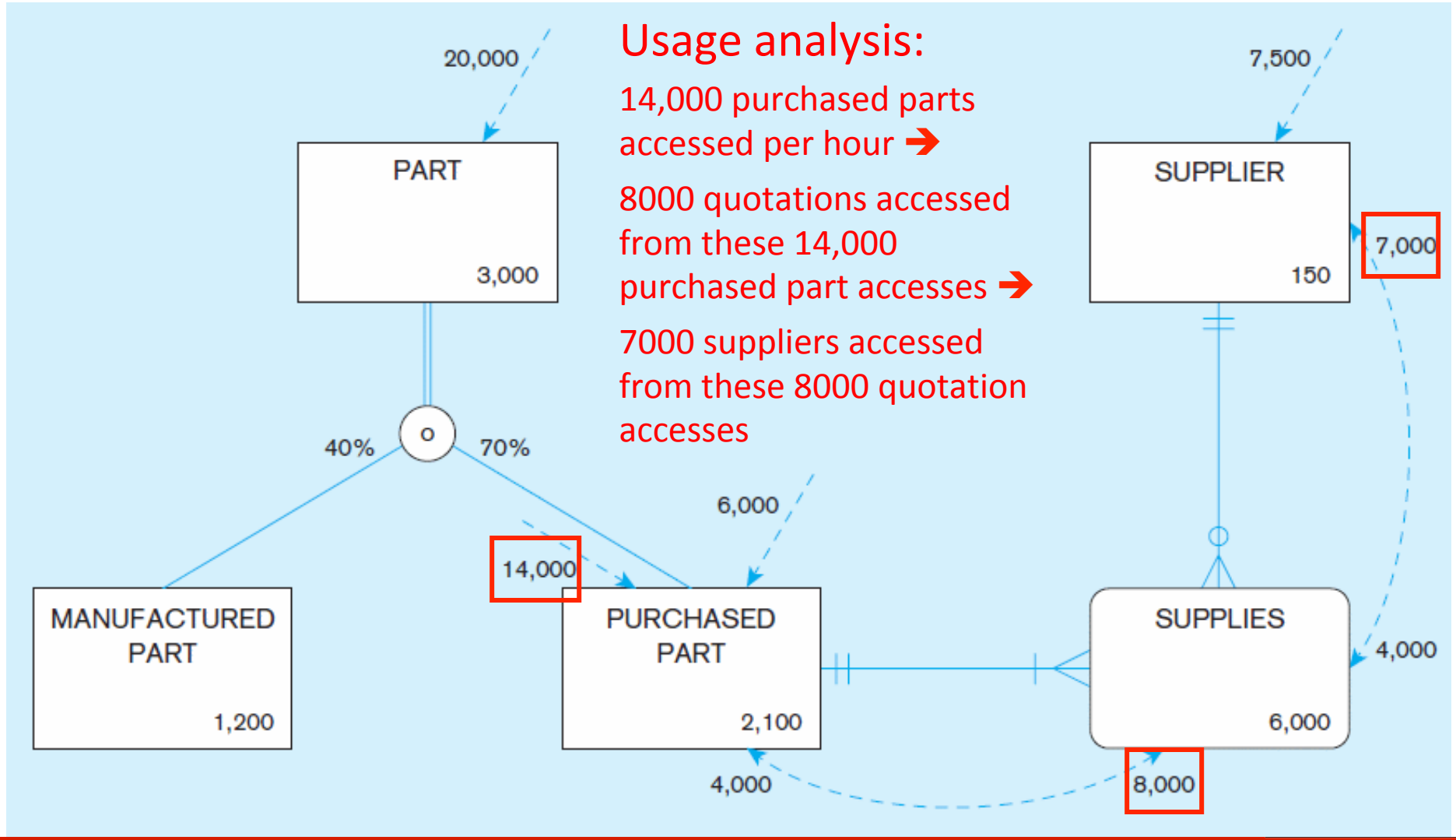
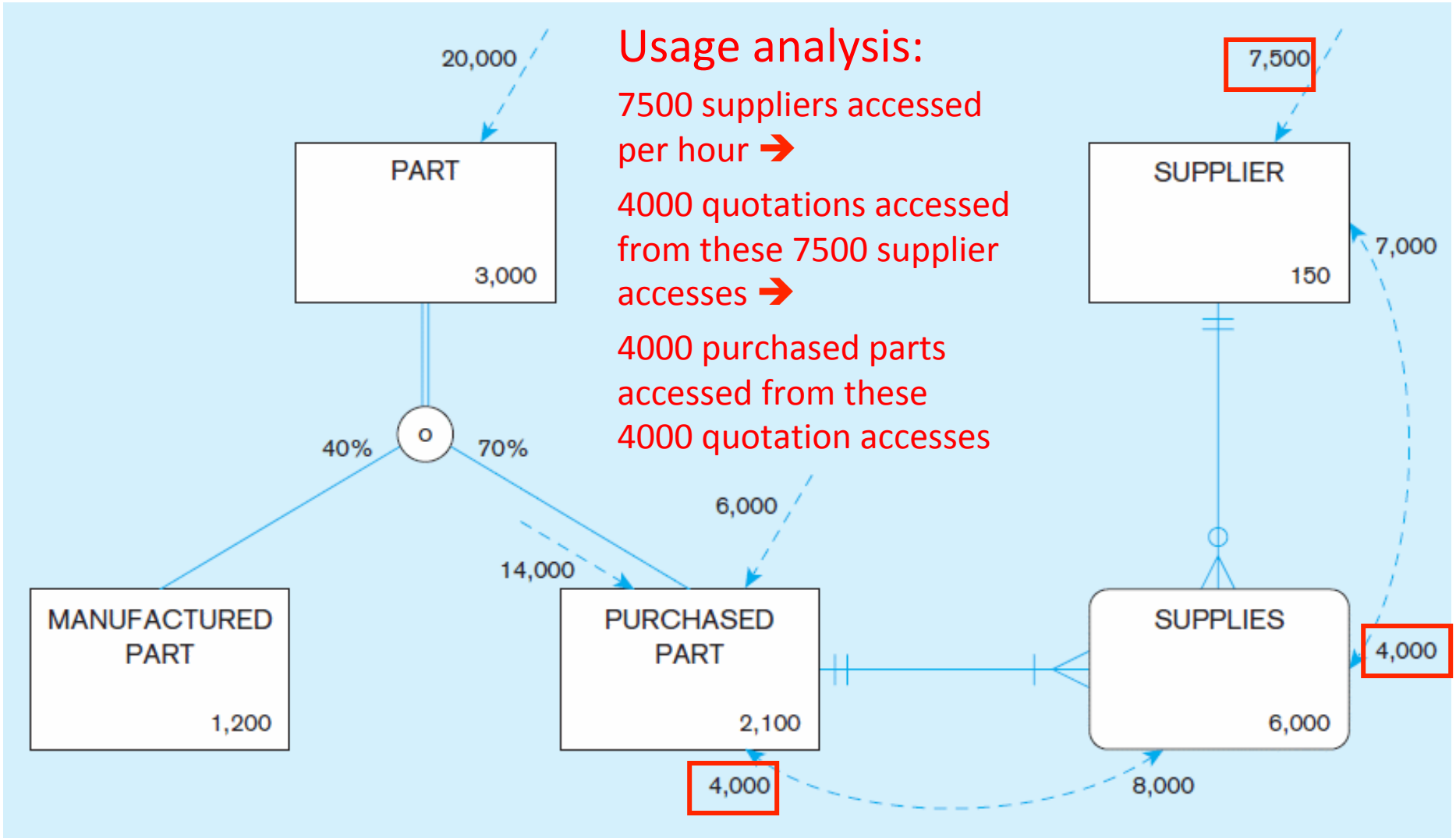


Figure 5-1: Composite Usage Map (Pine Valley Furniture Company)



Designing Fields

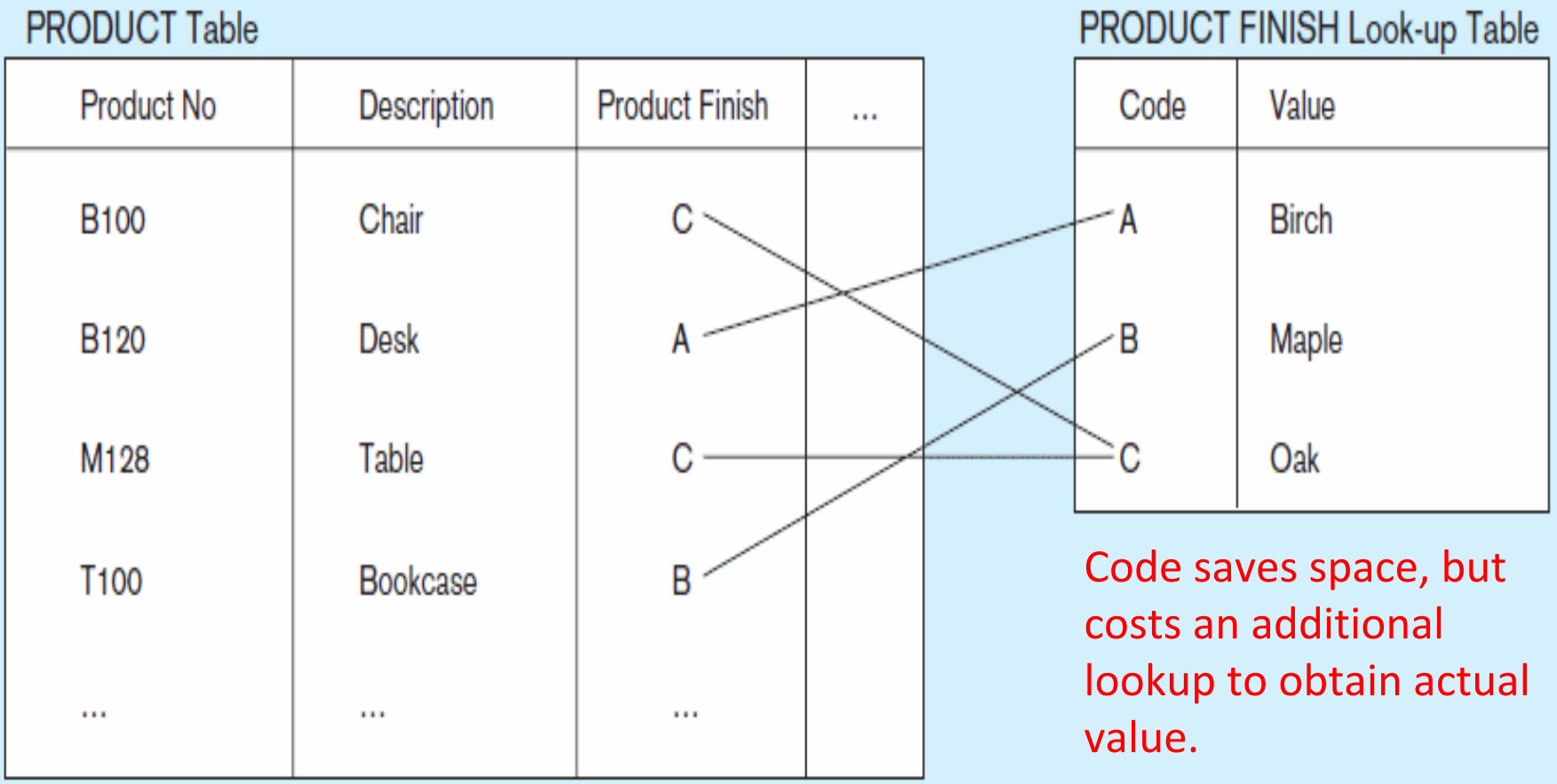
- **Field**: smallest unit of application data recognized by system software.
- **Field design**:
  - Choosing data type.
  - Coding, compression, encryption.
  - Controlling data integrity.

Table 5-1: Choosing Data Types

TABLE 5-1 Commonly Used Data Types in Oracle 11g

Data Type	Description
VARCHAR2 VARCHAR(n)	Variable-length character data with a maximum length of 4,000 characters; you must enter a maximum field length [e.g., VARCHAR2(30) specifies a field with a maximum length of 30 characters]. A string that is shorter than the maximum will consume only the required space.
CHAR	Fixed-length character data with a maximum length of 2,000 characters; default length is 1 character (e.g., CHAR(5) specifies a field with a fixed length of 5 characters, capable of holding a value from 0 to 5 characters long).
CLOB	Character large object, capable of storing up to (4 gigabytes – 1) * (database block size) of one variable-length character data field (e.g., to hold a medical instruction or a customer comment).
NUMBER INTEGER DECIMAL(p[,s]) REAL FLOAT	Positive or negative number in the range 10^{-130} to 10^{126}; can specify the precision (total number of digits to the left and right of the decimal point) and the scale (the number of digits to the right of the decimal point). For example, NUMBER(5) specifies an integer field with a maximum of 5 digits, and NUMBER(5,2) specifies a field with no more than 5 digits and exactly 2 digits to the right of the decimal point.
DATE	Any date from January 1, 4712 B.C., to December 31, 9999 A.D.; DATE stores the century, year, month, day, hour, minute, and second.
BLOB	Binary large object, capable of storing up to (4 gigabytes – 1) * (database block size) of binary data (e.g., a photograph or sound clip).

Figure 5-2: Example of a Code Look-Up Table (Pine Valley Furniture Company)



Field Data Integrity

- **Default value** – assumed value if no explicit value.
- **Range control** – allowable value limitations (constraints or validation rules).
- **Null value control** – allowing or prohibiting empty fields.
- **Referential integrity** – range control (and null value allowances) for foreign-key to primary-key match-ups.

Handling Missing Data

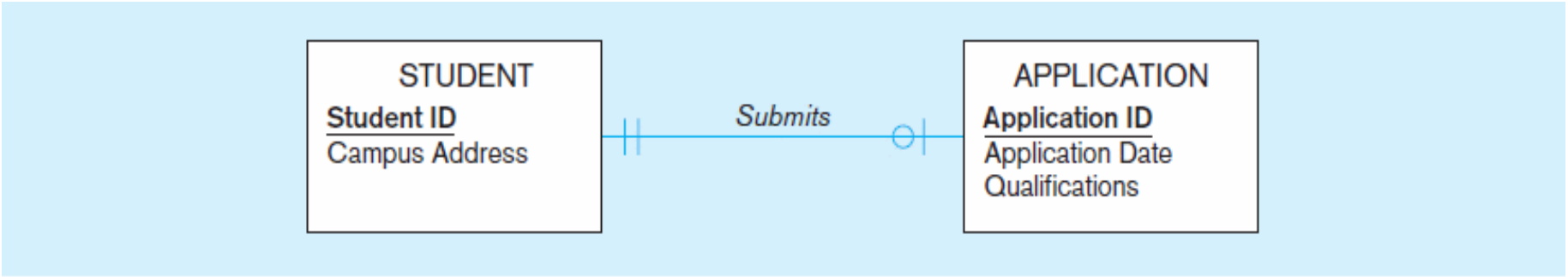
- Substitute an estimate of the missing value (e.g., using a formula).
- Construct a report listing missing values.
- In programs, ignore missing data unless the value is significant (sensitivity testing).

- Triggers can be used to perform these operations.

Denormalization

- Transforming **normalized** relations into **non-normalized** physical record specifications.
- Benefits:
  - Can improve performance (speed) by reducing number of table lookups (i.e. reduce number of necessary join queries).
- Costs (due to data duplication):
  - Wasted storage space.
  - Data integrity/consistency threats.
- Common denormalization opportunities:
  - One-to-one relationship (Fig. 5-3)
  - Many-to-many relationship with non-key attributes (associative entity) (Fig. 5-4)
  - Reference data (1:N relationship where 1-side has data not used in any other relationship) (Fig. 5-5)

Figure 5-3: Denormalization: Two Entities with One-to-One Relationship



Normalized relations:

STUDENT

Student ID	Campus Address

APPLICATION

Application ID	Application Date	Qualifications	Student ID

Denormalized relation:

STUDENT

Student ID	Campus Address	Application Date	Qualifications

and Application Date and Qualifications may be null

Figure 5-4: Denormalization: A Many-to-Many Relationship with Non-Key Attributes

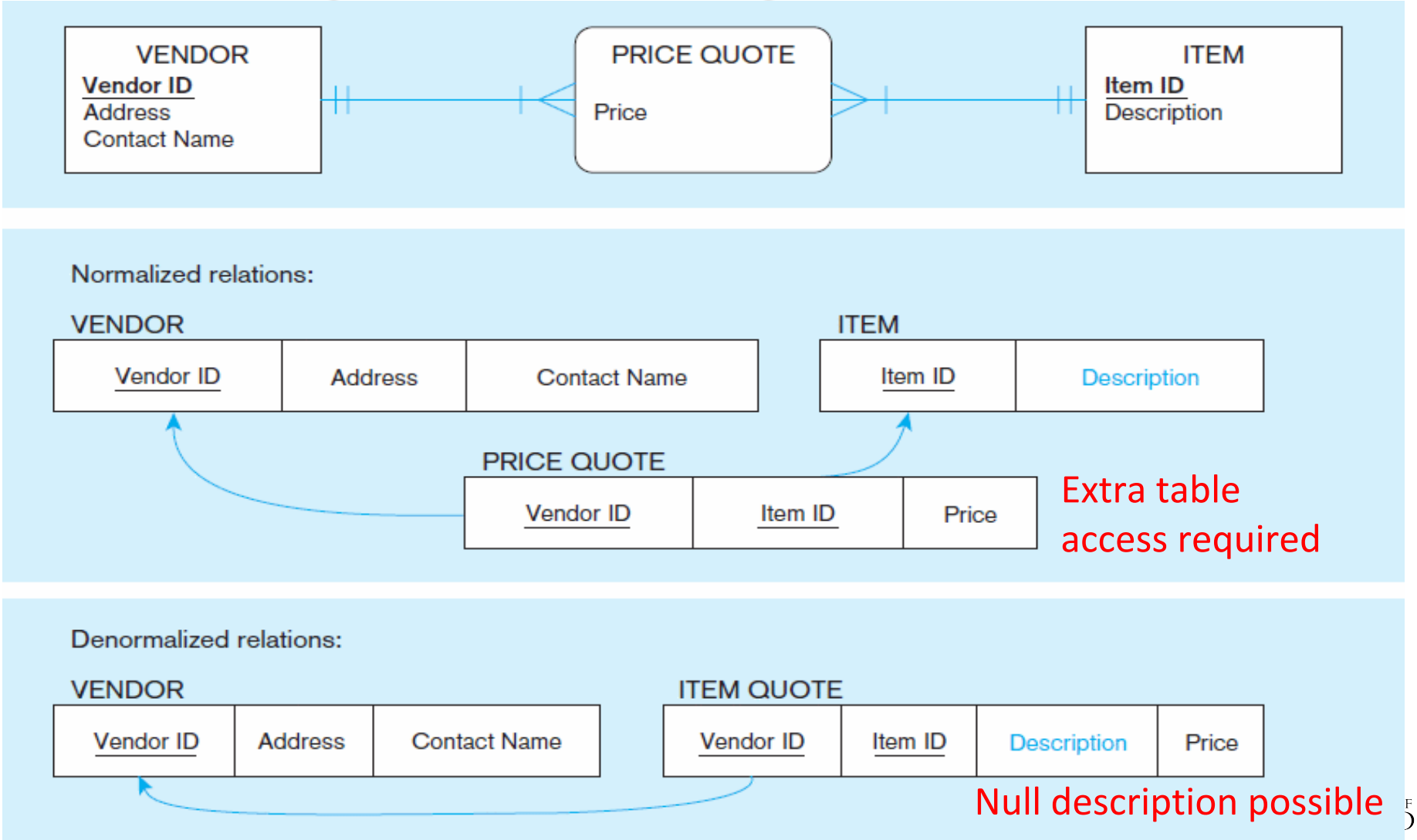
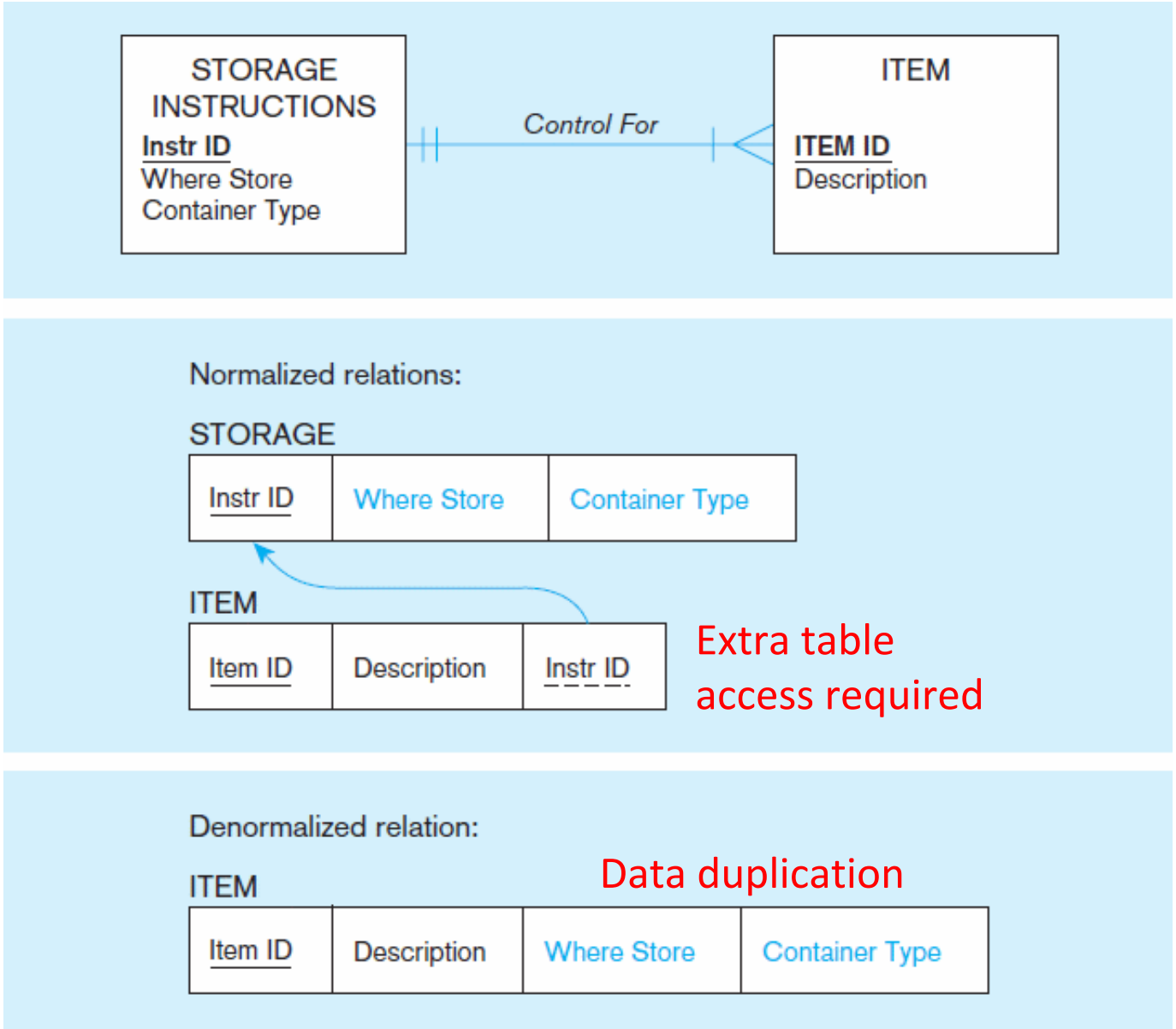


Figure 5-5: Denormalization: Reference Data



Denormalize with Caution

- Denormalization can:
  - Increase chance of errors and inconsistencies.
  - Reintroduce anomalies.
  - Force reprogramming when business rules change.
- Perhaps other methods could be used to improve performance of joins:
  - Organization of tables in the database (file organization and clustering).
  - Proper query design and optimization.

Partitioning

- **Horizontal Partitioning**: Distributing the rows of a logical relation into several separate tables:
  - Useful for situations where different users need access to different rows.
  - **Three types**: (1) Key Range Partitioning, (2) Hash Partitioning, or (3) Composite Partitioning.
- **Vertical Partitioning**: Distributing the columns of a logical relation into several separate physical tables:
  - Useful for situations where different users need access to different columns.
  - The primary key must be repeated in each file.
- **Combinations** of Horizontal and Vertical.

UNIVERSITY OF MARYLAND

Partitioning Pros and Cons

- Advantages of Partitioning:
  - **Efficiency**: Records used together are grouped together.
  - **Local optimization**: Each partition can be optimized for performance.
  - **Security**: data not relevant to users are segregated.
  - **Recovery and uptime**: smaller files take less time to back up.
  - **Load balancing**: Partitions stored on different disks, reduces contention.
- Disadvantages of Partitioning:
  - **Inconsistent access speed**: Slow retrievals across partitions.
  - **Complexity**: Non-transparent partitioning.
  - **Extra space or update time**: Duplicate data; access from multiple partitions.

Rules for Using Indexes

- Use on larger tables.
- Index the primary key of each table.
- Index search fields (fields frequently in WHERE clause).
- Fields in SQL ORDER BY and GROUP BY commands.
- When there are >100 values but not when there are <30 values.
- DBMS may have limit on number of indexes per table and number of bytes per indexed field(s).
- Be careful of indexing attributes with null values; many DBMSs will not recognize null values in an index search