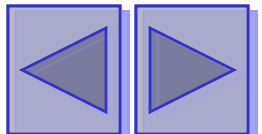
Modern Systems Analysis and Design

Chapter 4
Designing Databases

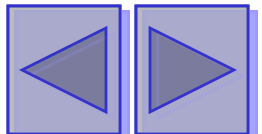
Learning Objectives

- ✔ Concisely define each of the following key database design terms: relation, primary key, normalization, functional dependency, foreign key, referential integrity, field, data type, null value, denormalization, file organization, index, and secondary key.
- ✔ Explain the role of designing databases in the analysis and design of an information system.
- ✔ Transform an entity-relationship (E-R) diagram into an equivalent set of well-structured (normalized) relations.



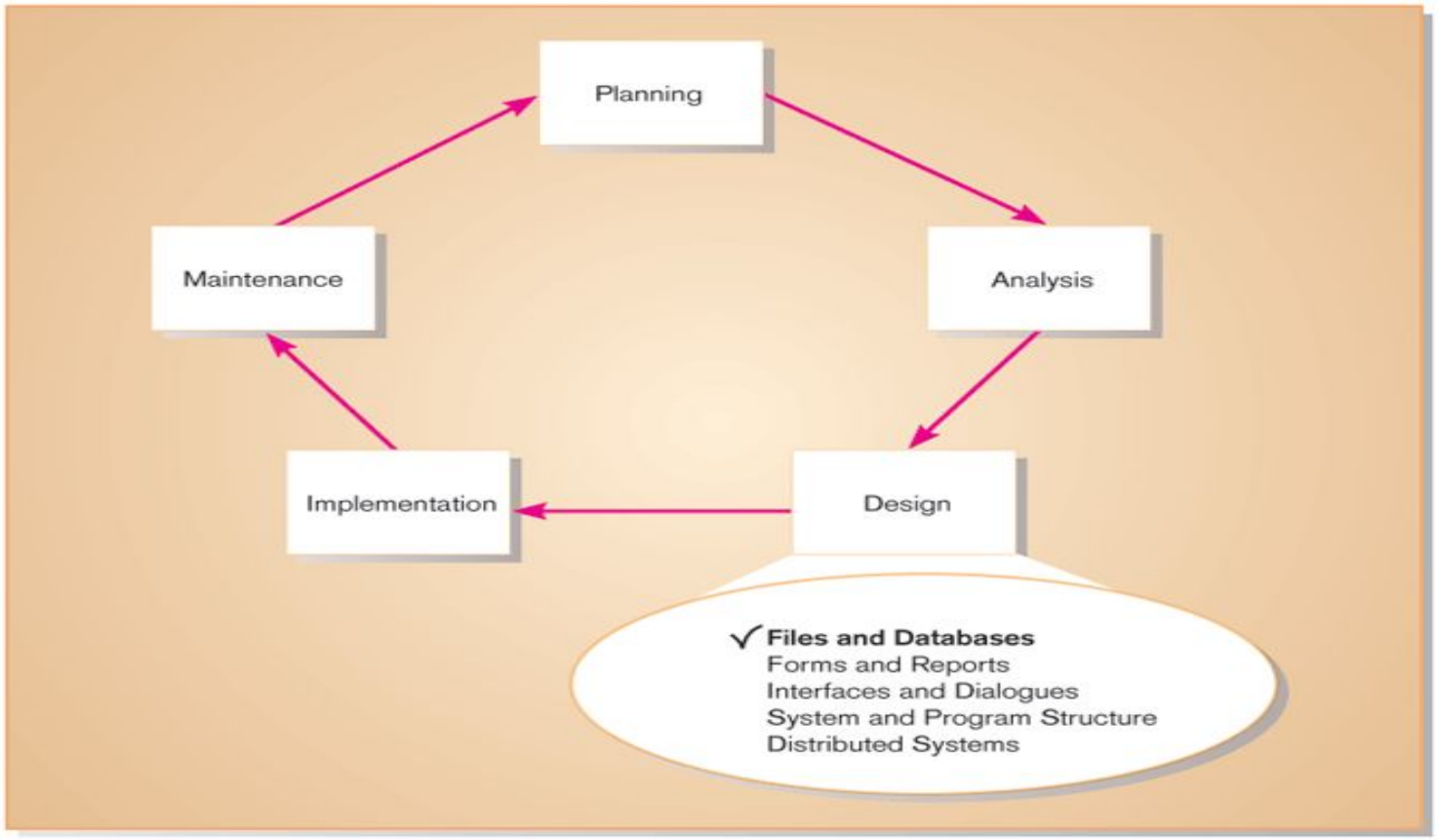
Learning Objectives

- ✔ Merge normalized relations from separate user views into a consolidated set of well-structured relations.
- ✔ Choose storage formats for fields in database tables.
- ✔ Translate well-structured relations into efficient database tables.
- ✔ Explain when to use different types of file organizations to store computer files.
- ✔ Describe the purpose of indexes and the important considerations in selecting attributes to be indexed



Introduction

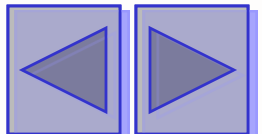
Figure 10-1 Systems development life cycle with design phase highlighted



Cis339

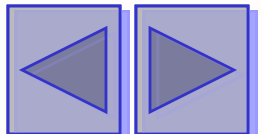
Database Design

- File and database design occurs in two steps.
- Develop a logical database model, which describes data using notation that corresponds to a data organization used by a database management system.
  - Relational database model.
- Prescribe the technical specifications for computer files and databases in which to store the data.
  - Physical database design provides specifications.
- Logical and physical database design in parallel with other system design steps.



The Process of Database Design

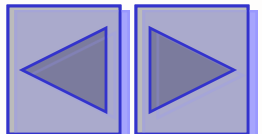
- Four key steps in logical database modeling and design:
  - Develop a logical data model for each known user interface for the application using normalization principles.
  - Combine normalized data requirements from all user interfaces into one consolidated logical database model (view integration).
  - Translate the conceptual E-R data model for the application into normalized data requirements.
  - Compare the consolidated logical database design with the translated E-R model and produce one final logical database model for the application



Process of Database Design

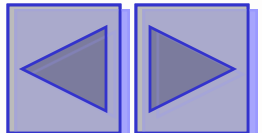
- **Physical Design**
  - **Based upon results of logical database design**
  - **Key decisions**
    1. **Choosing storage format (data type its length, number of decimal places.) for each attribute from the logical database model**
    2. **Grouping attributes from the logical database model into physical records.**
    3. **Arranging related records in secondary memory (hard disks and magnetic tapes) so that records can be stored, retrieved and updated rapidly**
    4. **Selecting media and structures for storing data to make access more efficient**

10.7



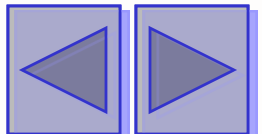
Deliverables and Outcomes

- **Logical database design**
  - Normalized relations are the primary deliverable.
  - Must account for every data element on a system input or output.
- Physical database design
  - Convert relations into database tables.
  - Programmers and database analysts code the definitions of the database.
  - Written in Structured Query Language (SQL).



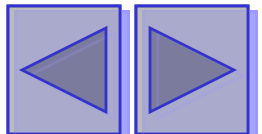
Relational Database Model

- Data represented as a set of related tables or relations
- Relation
  - A named, two-dimensional table of data. Each relation consists of a set of named columns and an arbitrary number of unnamed rows
  - Properties that distinguish them from other non relational tables:
    - Entries in cells are simple- it has a single value
    - Entries in columns are from the same set of values
    - Each row is unique
    - The sequence of columns can be interchanged without changing the meaning or use of the relation
    - The rows may be interchanged or stored in any sequence



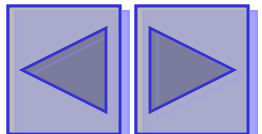
Well-Structured Relation and Primary key

- **Well-Structured Relation (**or **table)**
  - A relation that contains a minimum amount of redundancy;
  - Allows users to insert, modify, and delete the rows without errors or inconsistencies.
- Primary Key
  - An attribute whose value is unique across all occurrences of a relation.
- All relations have a primary key.
- This is how rows are ensured to be unique.
- A primary key may involve a single attribute or be composed of multiple attributes.



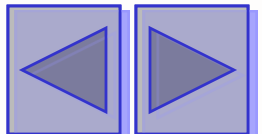
Normalization

- **Normalization**: the process of converting complex data structures into simple, stable data structures.
- First Normal From (1NF)
  - Unique rows, no multivalued attributes.
  - All relations are in 1NF.
- Second Normal Form (2NF)
  - Each nonprimary key attribute is identified by the whole key (called full functional dependency). **In figure 10-7 both the EMP ID and Course can identify the Date-Completed.**
- Third Normal Form (3NF)
  - Nonprimary key attributes do not depend on each other (i.e. no transitive dependencies).
- The result of normalization is that every nonprimary key attribute depends upon the whole primary key.



Functional Dependencies

- **Functional Dependency**
  - A particular relationship between two attributes.
  - For a given relation, attribute B is functionally dependent on attribute A if, for every valid value of A, that value of A uniquely determines the value of B.
  - The functional dependence of B on A is represented by A→B.
  - **examples are order number -> order date, or invoice number-> invoice date and order number.**
- Functional dependency is not a mathematical dependency.
- Instances (or sample data) in a relation do not prove the existence of a functional dependency.
- Knowledge of problem domain is most reliable method for identifying functional dependency.



Second Normal Form

- **Second Normal Form (2NF) is**
  - **A relation is in second normal form (2NF) if any of the following conditions apply:**
    - **The primary key consists of only one attribute**
    - **No non-primary key attributes exist in the relation**
    - **Every non-primary key attribute is functionally dependent on the full set of primary key attributes,** ***in figure 10-6 EMPLOYEE2 is not 2NF because date completed depends on EMP_id and Course, but name, salary, and dept depend on EMP_id not Course .***

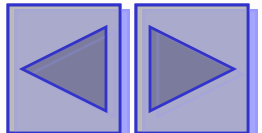
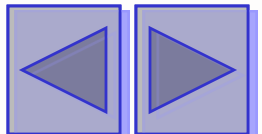


Figure 9.6 Relation with Redundancy

EMPLOYEE2

Emp_ID	Name	Dept	Salary	Course	Date_Completed
100	Margaret Simpson	Marketing	$42,000	SPSS	6/19/2002
100	Margaret Simpson	Marketing	42,000	Surveys	10/7/2002
140	Alan Beeton	Accounting	39,000	Tax Acc	12/8/2002
110	Chris Lucero	Info Systems	41,500	SPSS	1/12/2002
110	Chris Lucero	Info Systems	41,500	C++	4/22/2002
190	Lorenzo Davis	Finance	38,000	Investments	5/7/2002
150	Susan Martin	Marketing	38,500	SPSS	6/19/2002
150	Susan Martin	Marketing	38,500	TQM	8/12/2002



Second Normal Form

- **Conversion to second normal form (2NF)**
  - **To convert a relation into 2NF, decompose the relation into new relations using the attributes, called determinants, that determine other attributes**
  - **The determinants become the primary key of the new relation, Converting figure 10-6 to 2NF we need to split it to Figure 10-5 and 10-7.**

Third Normal Form

- **Third Normal Form (3NF) is**
  - **A relation is in third normal form (3NF) if it is in second normal form (2NF) and there are no fun transitive dependencies between two (or more) nonprimary key attributes.**
  - **See figure 10-9 SALES (customer_id, name, salesperson, region)** ***where region is functionally dependent on salesperson*** **and** ***salesperson is functionally dependent on customer_id-.*** **See how the transitive dependencies removed by creating two relations one named SALES1 (customer id, customer name, salesperson) and other is SPERSON (salesperson, region)**

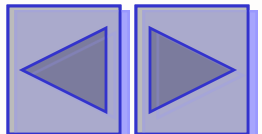


Figure 9.9a Removing Transitive Dependencies — Relation with Transitive Dependency

SALES

Customer_ID	Customer_Name	Salesperson	Region
8023	Anderson	Smith	South
9167	Bancroft	Hicks	West
7924	Hobbs	Smith	South
6837	Tucker	Hernandez	East
8596	Eckersley	Hicks	West
7018	Arnold	Faulb	North

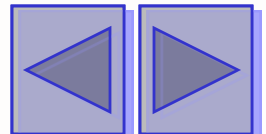
Figure 9.9b Removing Transitive Dependencies — Relations in 3NF

SALES1

Customer_ID	Customer_Name	Salesperson
8023	Anderson	Smith
9167	Bancroft	Hicks
7924	Hobbs	Smith
6837	Tucker	Hernandez
8596	Eckersley	Hicks
7018	Arnold	Faulb

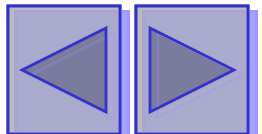
SPERSON

Salesperson	Region
Smith	South
Hicks	West
Hernandez	East
Faulb	North



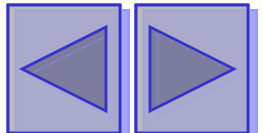
Foreign Keys and referential integrity

- **Foreign Key**
  - **An attribute that appears as a non-primary key attribute in one relation and as a primary key attribute (or part of a primary key) in another relation- The FK must satisfy referential integrity.**
- **Referential Integrity**
  - **An integrity constraint specifying that the value (or existence) of an attribute in one relation depends on the value (or existence) of the same attribute in another relation (*the value of Salesperson in each row of Sales1 is limited to only the current values of current Salesperson in the Sperson table*)**



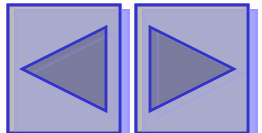
Transforming E-R Diagrams into Relations

- It is useful to transform the conceptual data model (E-R) into normalized relations, then merge all in one final, consolidated set of relation which can be accomplished by the following steps:
  - Represent entities
  - Represent relationships
  - Normalize the relations
  - Merge the relations



Transforming E-R Diagrams into Relations

- **Represent Entities**
  - **Each regular entity type in an E-R is transformed into a relation**
  - **The identifier of the entity type becomes the primary key of the corresponding relation**
  - **The primary key must satisfy the following two conditions**
    a. **The key should be none-redundant, which mean no attribute in the key can be deleted without destroying its unique identification**
    b. **The value of the key must uniquely identify every row in the relation**
  - The entity type label is translates into a relation name.



E.g

Figure 9.10a Transforming an Entity Type to a Relation — E-R Diagram

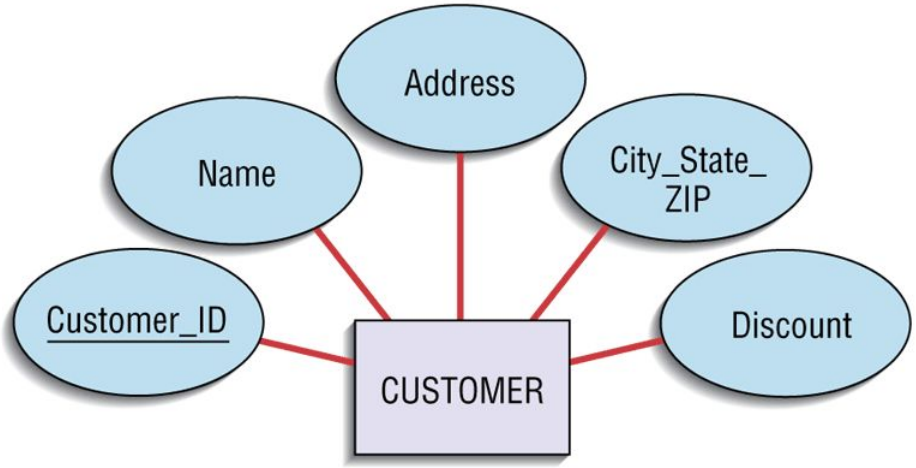
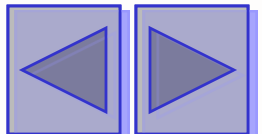


Figure 9.10b Transforming an Entity Type to a Relation — Relation

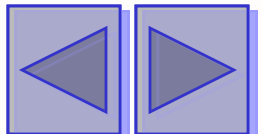
CUSTOMER

Customer_ID	Name	Address	City_State_ZIP	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%



Transforming E-R Diagrams into Relations

- **Represent Relationships**
  - **Binary 1:N Relationships**
    - Add the primary key attribute (or attributes) of the entity on the one side of the relationship as a foreign key in the relation on the many side
    - The one side *migrates* to the many side, see figure 10-11 where CUSTOMER ID which it is the primary key in CUSTOMER becomes a FK to the relation ORDER
  - **Binary or Unary 1:1**
    - Three possible options
      a. Add the primary key of A as a foreign key of B
      b. Add the primary key of B as a foreign key of A
      c. Both of the above



- E.g

Figure 9.11a Representing a 1:N Relationship — E-R Diagram

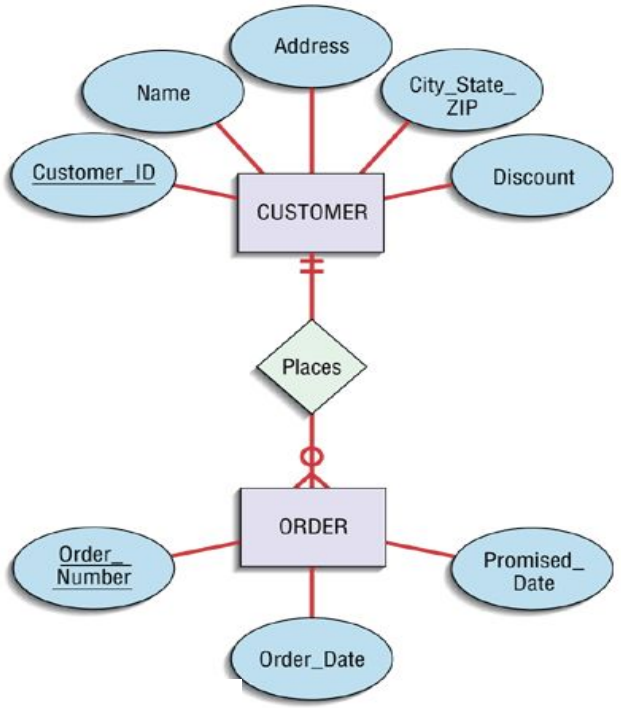


Figure 9.11b Representing a 1:N Relationship — Relations

CUSTOMER

Customer_ID	Name	Address	City_State_ZIP	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

ORDER

Order_Number	Order_Date	Promised_Date	Customer_ID
57194	3/15/0X	3/28/0X	6390
63725	3/17/0X	4/01/0X	1273
80149	3/14/0X	3/24/0X	6390

Transforming E-R Diagrams into Relations

Represent Relationships (continued)

- **Binary and Higher M:N relationships**
  - Create another relation and include primary keys of all relations as primary key (composite) of new relation
- **Unary 1:N Relationships**
  - Relationship between instances of a single entity type
  - Utilize a recursive foreign key
    - A foreign key in a relation that references the primary key values of that same relation ex. EMPLOYEE(emp_id, Name, Birthdate, Manager_id)
- **Unary M:N Relationships**
  - Create a separate relation
  - Primary key of new relation is a composite of two attributes that both take their values from the same primary key ex. ITEM(item_no, name, Cost) and ITEM_BILL(item_no, component_id, Quantity)

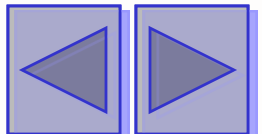


Figure 9.12b Representing an M:N Relationship — Relations

ORDER

Order_Number	Order_Date	Promised_Date
61384	2/17/2002	3/01/2002
62009	2/13/2002	2/27/2002
62807	2/15/2002	3/01/2002

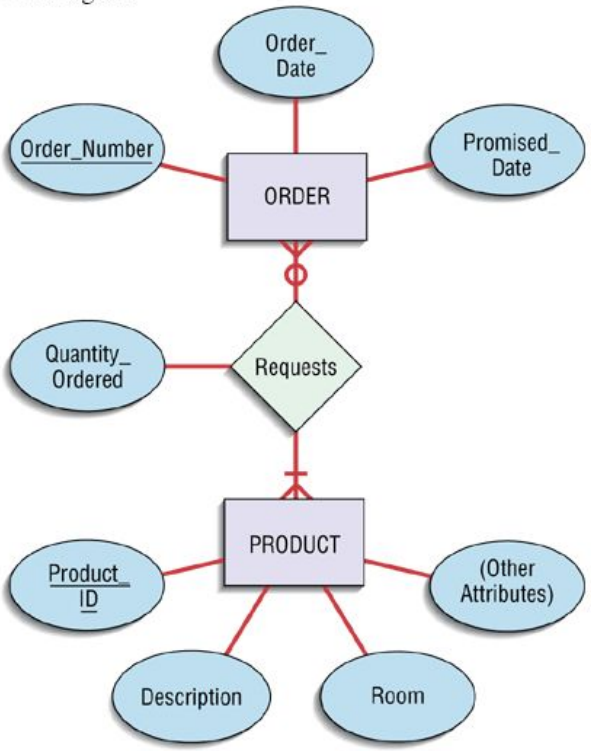
ORDER LINE

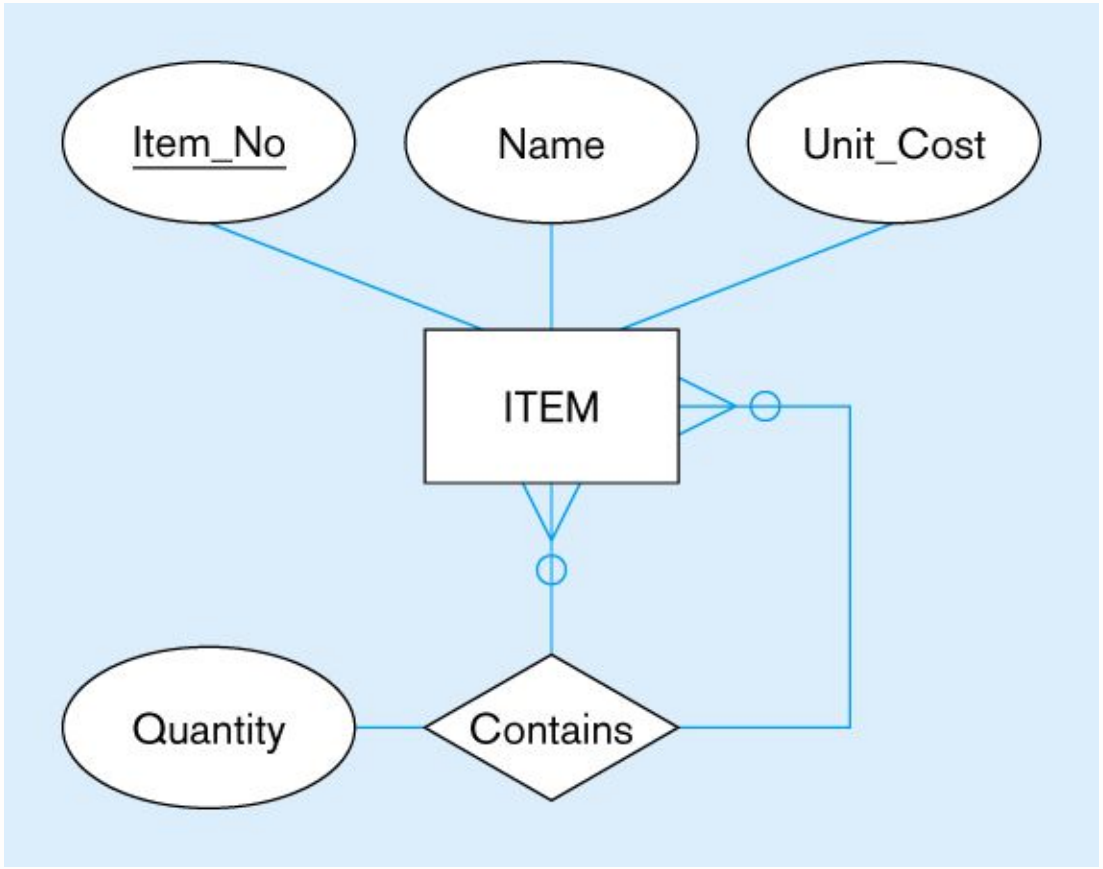
Order_Number	Product_ID	Quantity_ Ordered
61384	M128	2
61384	A261	1

PRODUCT

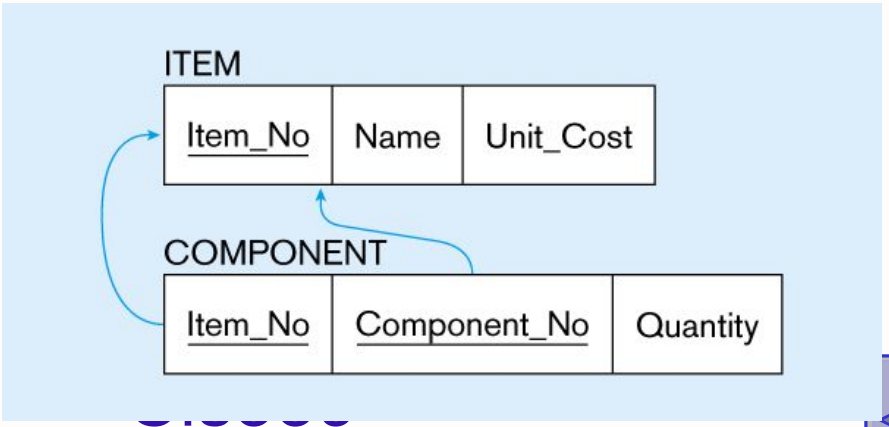
Product_ID	Description	(Other Attributes)
M128	Bookcase	—
A261	Wall unit	—
R149	Cabinet	—

Figure 9.12a Representing an M:N Relationship — E-R Diagram



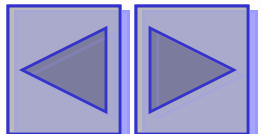


Mapping a unary M:N relationship



Merging Relations

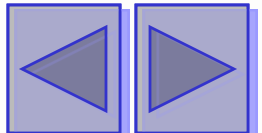
- **Merging Relations**
  - Purpose is to remove redundant relations,
  - when modeling user interfaces, ex Employee1(EMP_id, Name, Address, Phone) and Employee2(Emp_id, Name, Address, Jobcode). Merge them in Employye(EMP_id, Name, Adress, Phone, Jobcoe)
  - View Integration Problems
    - Synonyms (student-id, matriculation-no)
      - Two different names used for the same attribute
      - When merging, get agreement from users on a single, standard name
    - Homonyms (Account which it could be saving account, checking account..)
      - A single attribute name that is used for two or more different attributes
      - Resolved by creating a new name
    - Dependencies between nonkeys (see next slide for example)
      - Dependencies may be created as a result of view integration
      - In order to resolve, the new relation must be normalized



For example consider the following:
student1(student_id,major)
student2(student_id,advisor)
we can merge the two relations in one:
student(student_id,major, advisor)

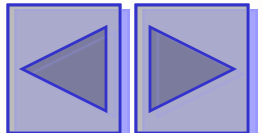
Now what if there is an exactly one advisor for the major, then student is 2NF but not 3NF, because it contains functional dependencies . In order to make it 3NF we need to do:
student(student_id,major)
major advisor(major,advisor)

- Class/ Subclass, relationship may be hidden in user views for example
  patient1(patient_id,Name, Address)
  patient2(patient_id,Room_number)
  Initially it appears that we can emerge these two relation, However there are different type of patient, inpatient and outpatient.
  Patient(patient_id,Name, Address)
  inpatient(Patient_id,Room_no)
  outpatient(patient_id,Date_treated)



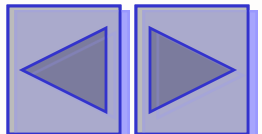
Physical File and Database Design

- **The following information is required:**
  - **Normalized relations**
  - **Definitions of each attribute**
  - **Descriptions of where and when data are used, entered, retrieved, deleted and updated**
  - **Expectations or requirements for response time and data integrity**
  - **Descriptions of the technologies used for implementing the files and database**



Designing Fields

- **Field**
  - The smallest unit of named application data recognized by system software
  - Each attribute from each relation will be represented as one or more fields
- **Choosing data types**
  - Data Type
    - A coding scheme recognized by system software for representing organizational data
  - Four objectives
    - Minimize storage space
    - Represent all possible values of the field
    - Improve data integrity of the field
    - Support all data manipulations desired on the field
  - **Calculated field:** a field that can be derived from other database fields. It is common for an attribute to be mathematically related to other data. The calculate value is either stored or computed when it is requested.

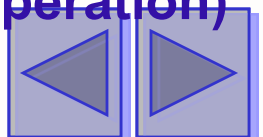


Methods of Controlling Data Integrity

- **Default Value**
  - A value a field will assume unless an explicit value is entered for that field
- **Range Control**
  - Limits range of values which can be entered into field
- **Referential Integrity**
  - An integrity constraint specifying that the value (or existence) of an attribute in one relation depends on the value (or existence) of the same attribute in another relation (foreign key must satisfy it)
- **Null Value**
  - A special field value, distinct from 0, blank, or any other value, that indicates that the value for the field is missing or otherwise unknown

Designing Physical Tables

- **Relational database is a set of related tables**
- **Physical Table**
  - **A named set of rows and columns that specifies the fields in each row of the table**
- **Design Goals**
  - **Efficient use of secondary storage (disk space)**
    - **Disks are divided into units (pages) that can be read or written in one machine operation**
    - **Space is used most efficiently when the physical length of a table row divides close to evenly with storage unit**
  - **Efficient data processing**
    - **Data are most efficiently processed when stored next to each other in secondary memory (minimizing the I/O operation)**



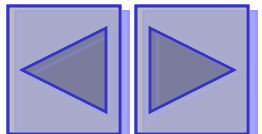
Designing Physical Tables

Denormalization

- The process of combining normalized relations into physical tables based on affinity(relationship) of use of rows and fields.

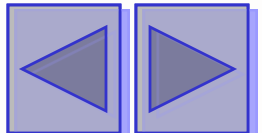
Arranging Table Rows

- Physical File
  - A named set of table rows stored in a contiguous section of secondary memory
- Each table may be a physical file or the whole database may be in one file, depending on database management software utilized



Designing Physical Tables

- **File Organization**
  - **A technique for physically arranging the records of a file.**
  - **Objectives for choosing file organization**
    1. **Fast data retrieval**
    2. **High throughput for processing transactions**
    3. **Efficient use of storage space**
    4. **Protection from failures or data loss**
    5. **Minimizing need for reorganization**
    6. **Accommodating growth**
    7. **Security from unauthorized use**



Designing Physical Tables

- **Types of File Organization**
  - **Sequential File Organization**
    - The rows in the file are stored in sequence according to a primary key value
    - Updating and adding records may require rewriting the file
  - **Indexed File Organization**
    - The rows are stored either sequentially or non-sequentially and an index is created that allows software to locate individual rows.
  - **Hashed File Organization**
    - A file organization in which the addresses of each row is determined using an algorithm.