1. Database Modeling Introduction

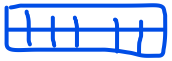
layer → Ⓡ → NoSQL (SNS, 다양)
(계층 (테이블 (그래프 문서 분산위주

Background

- Relational database system
  - Dominant database technology for business enterprises
- Relational database design (E-R 모델링 → Ⓡ 모델링)
  - has evolved from an art to a science 경력과경답
  - partially implementable as a set of software design aids
    - ERwin Data Modeler or Rational Rose with UML

데이터 아키텍처

- Logical design 스키마



  - the structure of basic data relationships and their definition in a particular database system
  - the domain of application designers → 일맞게
  - work effectively with tools such as ERwin Data Modeler or Rational Rose with UML
- Physical design
  - the creation of efficient data storage and retrieval mechanisms on the computing platform
  - domain of the database administrator(DBA), DBMS
    - Today's DBAs have a variety of vendor-supplied tools
- This book is devoted to the logical design

Data item, Record, File

- data item (필드) Atomic
  의미
  - the smallest named unit of data that has meaning in the real world
  - last name, first name, street address, ID number
- Record (튜플)
  개체
  - A group of related data items treated as a single unit by an application
  - order, salesperson, customer, product, and department
- File (테이블이 다수)
  - a collection of records of a single type
- In a relational database
  - a data item is called a *column* or *attribute*
  - a record is called a *row* or *tuple*
  - a file is called ~~a~~ *table*

File $\xrightarrow{\text{추상화}}$

Database

- A more complex object
- A collection of interrelated 관계 stored data that serves the needs of multiple users within one or more organizations 복수
  - that is, interrelated collections of many different types of tables

- Rather than files < DB
  - 공통 greater availability to a diverse set of users → view와 목표에 따라 다름
  - integration of data for easier access to and updating of complex transactions → 불일치X
  - less redundancy of data → 중복X

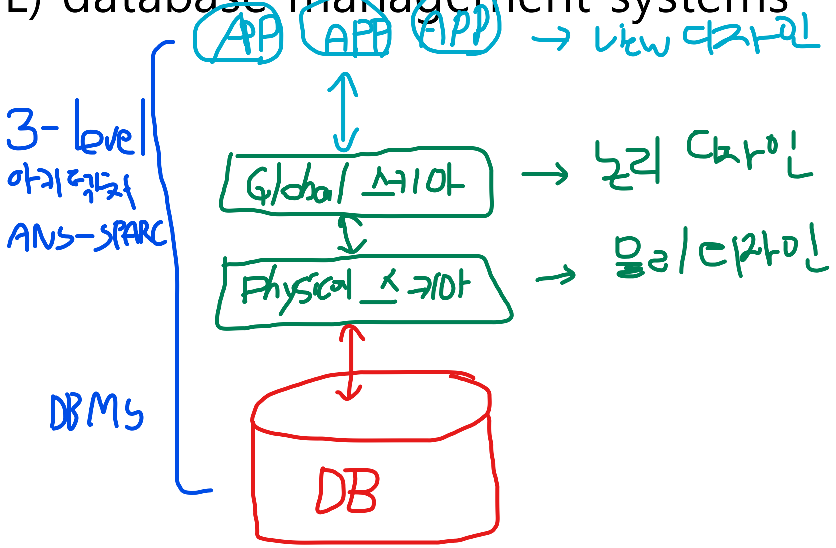
DBMS

- A generalized software system for manipulating databases
- Supports
  - Logical view (schema, subschema)
  - Physical view (access methods, data clustering)
  - Data definition language; DDL
  - Data manipulation language; DML
  - Important utilities
    - Transaction management ACID
    - Concurrency control 병행
    - Data integrity 무결성 - C~~R~~UD
    - Crash recovery 회복 - Log
    - Security 보안

Data Independence

- *Data independence* 스키마구조가 변해도 다른 곳 영향X
  - The ability to make changes in either the logical or physical structure of the database without requiring reprogramming of application programs
  - It makes database conversion and reorganization much easier
- Relational database systems 비고 니상
  - Provide a greater degree of data independence than the earlier hierarchical and network (CODASYL) database management systems

NOSQL – 분석용



The Database Life Cycle

- The basic steps involved in designing a global schema of the logical database
- Once the design is completed, the life cycle continues with database implementation and maintenance

I. Requirements analysis

II. Logical design

  a. Conceptual data modeling

  b. View integration → 부서별 다른 모습 통합

  c. Transformation of the conceptual data model to SQL tables

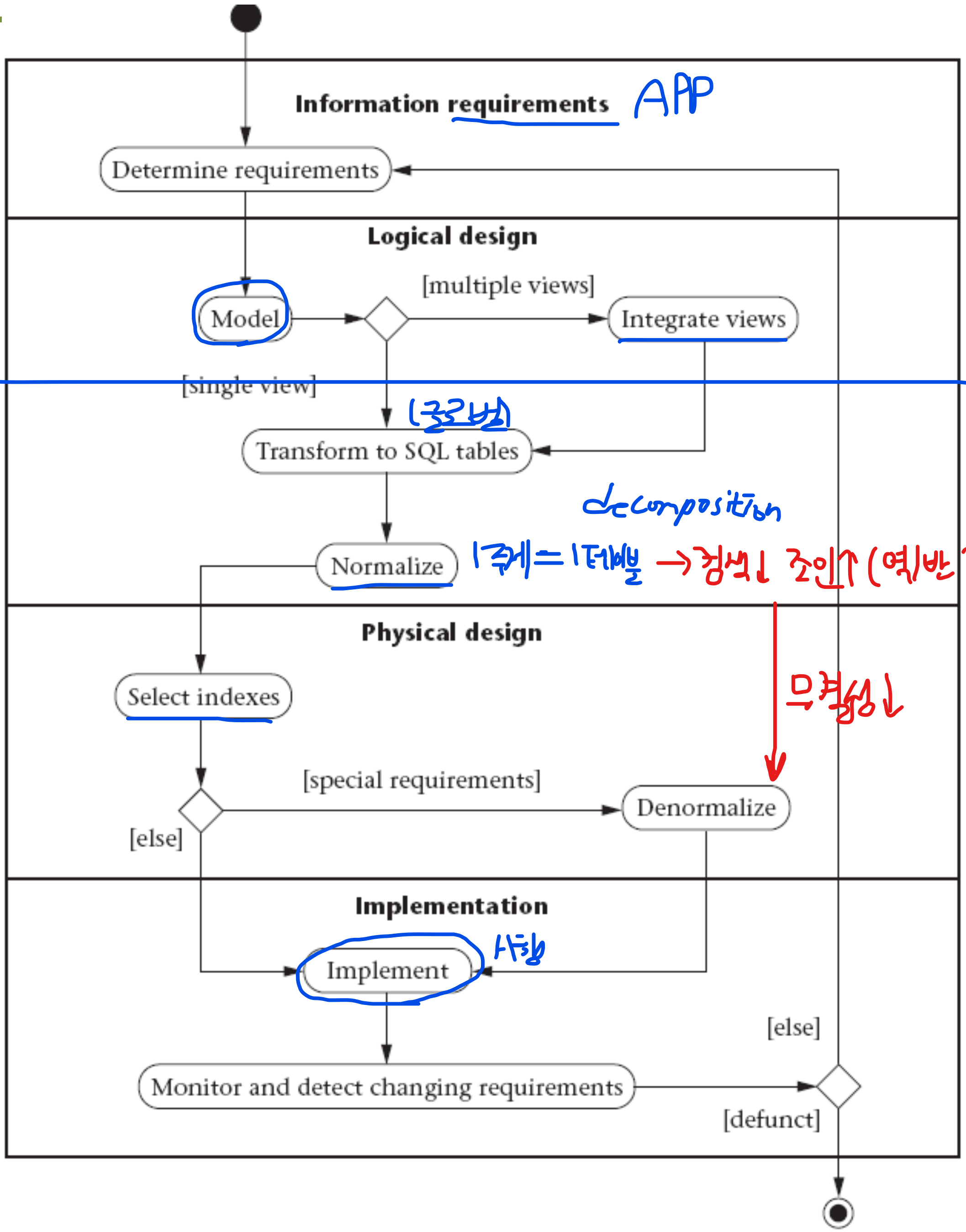
  d. Normalization of tables

III. Physical design

IV. Database implementation, monitoring, and modification

SQL-구조질의 언어

The Database Life Cycle



Requirements analysis – 사용자의 요구사항(정보시스템)

- Determined by interviewing both the producers and users of data
- Using the information to produce a formal requirements specification
  정형화
  - That specification includes the data required for processing, the natural data relationships, and the software platform for the database implementation

Step I Requirements Analysis (reality)

  - Formulated in the mind of the end user during the interview process
    세세하게 사용자관점 → E-R 모델링(win)으로 쉽게 이해

Logical Design

- The *global schema*, a conceptual data model diagram
- Shows all the data and their relationships
- Developed using techniques such as ER or UML
- The data model constructs must ultimately be transformed into normalized (global) relations
- The global schema development methodology is the same for either a distributed or centralized database

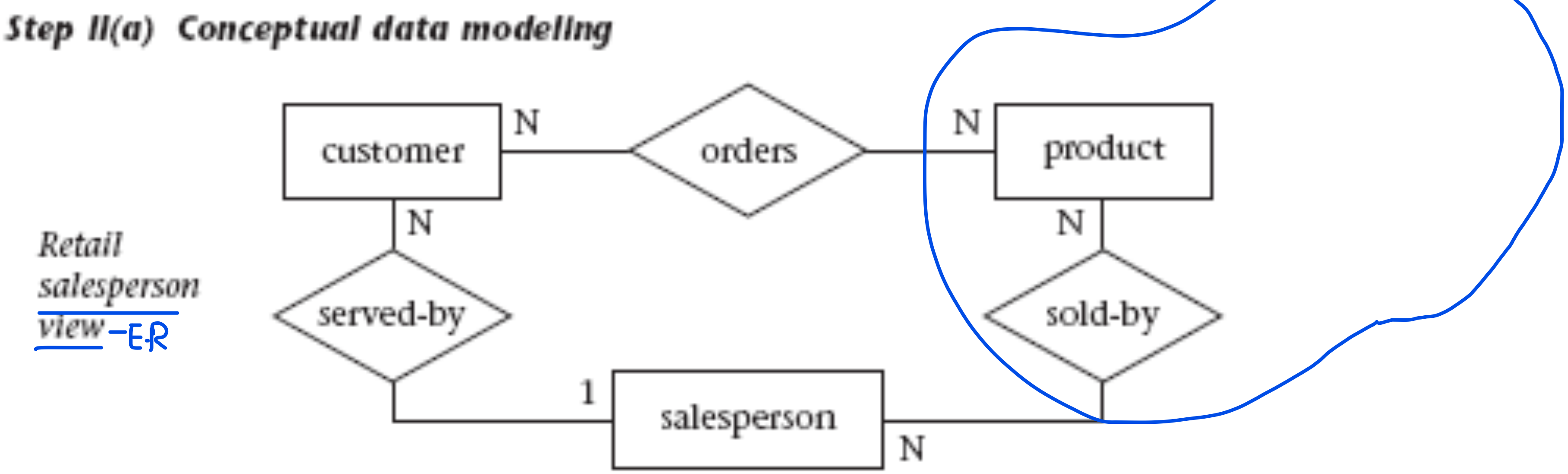
  은행 지점들 나누어짐

- Conceptual data modeling → E-R
- View integration → 글로벌 스키마
- Transformation of the conceptual data model to SQL tables
- Normalization of tables → 정규화

Conceptual data modeling

- The data requirements are analyzed and modeled using an ER or UML diagram
  - Including semantics for optional relationships, ternary relationships, supertypes, and subtypes

Product view – E-R

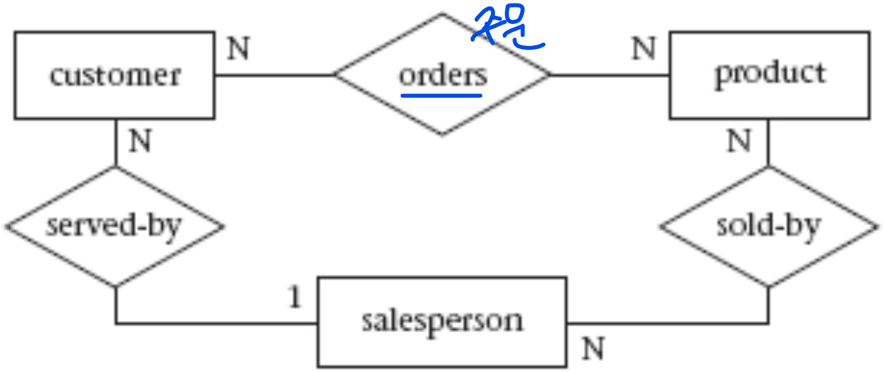


Step II(a) Conceptual data modeling

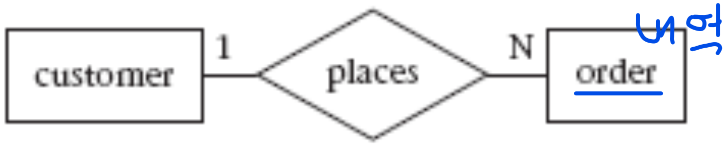
View Integration

- When the design is large and more than one person is involved in requirements analysis → multiple views of data and relationships result
- To eliminate redundancy and inconsistency from the model, these views must eventually be "rationalized" → 정리, 합리화
  - resolving inconsistencies due to variance in taxonomy, context, or perception
- Requires the use of ER semantic tools such as identification of synonyms, aggregation, and generalization.

Retail salesperson view

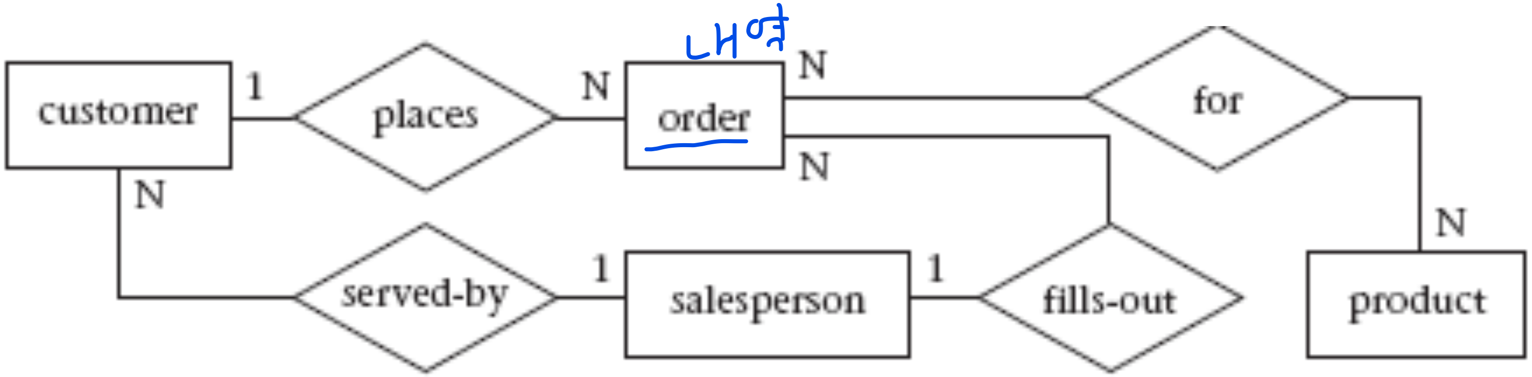


Customer view



View Integration

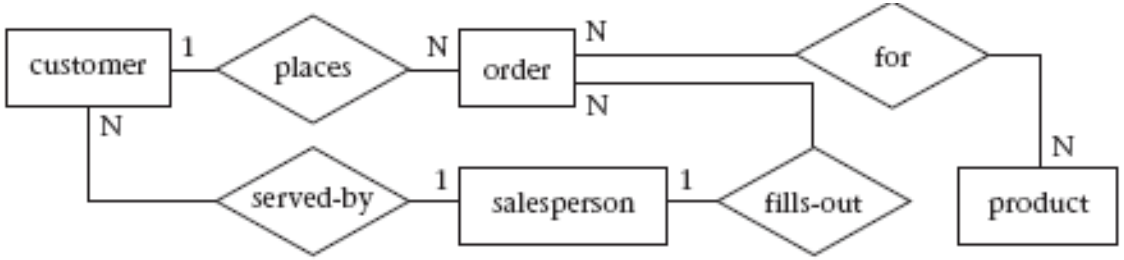
Integration of retail salesperson's and customer's views



Transformation of the conceptual data model to SQL tables

- Each relationship and its associated entities are transformed into a set of DBMS-specific candidate relational tables
- Redundant tables are eliminated as part of this process

Integration of retail salesperson's and customer's views



E-R WIN

Customer

cust-no	cust-name	. . .

Product

prod-no	prod-name	qty-in-stock

Salesperson

sales-name	addr	dept	job-level	vacation-days

Order

order-no	sales-name	cust-no

Order-product

order-no	prod-no

```
create table customer
    (cust_no integer,
    cust_name char(15),
    cust_addr char(30),
    sales_name char(15),
    prod_no integer,
    primary key (cust_no),
    foreign key (sales_name)
        references salesperson
    foreign key (prod_no)
        references product);
```

Oracle
SQL sever

그러나 설계가 이상하면
정규화가 필요

인덱스
클러스터 ← 검색 ∝ 갱신
파티션
join
역정규화

Normalization of tables

1table = 1주제 = 1결정자

- Functional dependencies (FDs) between tables $X \rightarrow Y$
  - Derived from
    - The conceptual data model diagram and
    - The semantics of data relationships in the requirements analysis
  - Represent the dependencies among data elements that are unique identifiers (keys) of entities
- Functional dependencies (FDs) within tables
  - Represent the dependencies among key and nonkey attributes within entities
  - Can be derived from the requirements specification
- Redundancies in the data in normalized candidate tables are analyzed further for possible elimination, with the constraint that data integrity must be preserved BCNF = F^{C} (F^{+} 모든추론)

주제의 구성 spec
<학생>
20173010 → 이름, 컴공, 3.5

반복

유일성
super
candidate − A, C, E
Primary
Alternate 대체

T A B C D E F . .

A → C → A
C → c → E
A → A → C
E → A → E
C → E → C
E → A

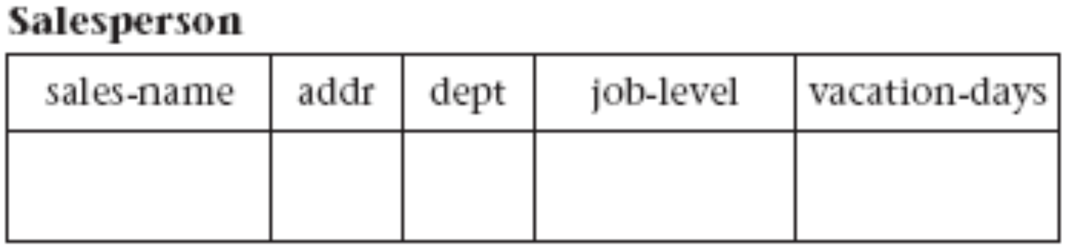
모든결정자가 후보키

(BCNF)
"정규화" ← 삽입 갱신 삭제

X ← 이상
(DB거의안) anomaly

Normalization of tables

설계
E-R



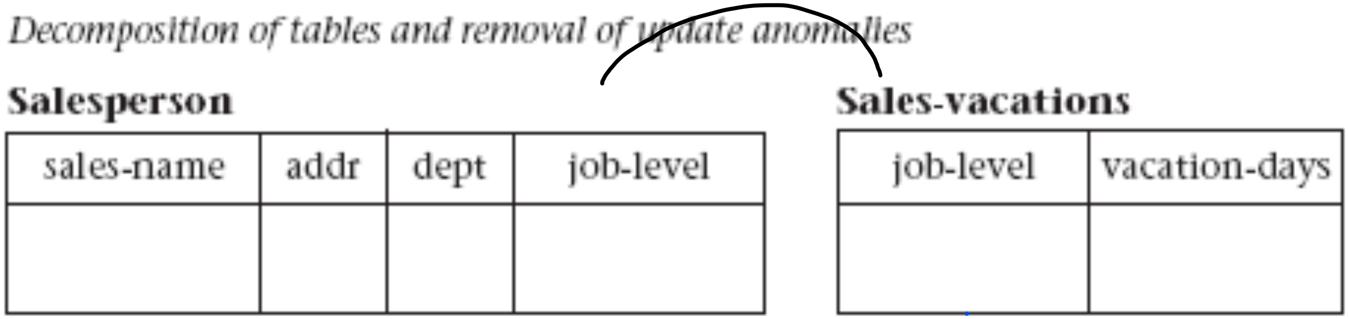
Salesperson

sales-name	addr	dept	job-level	vacation-days

조회시간 상승↓
(일관성, 무결성)

D
B

Join



Decomposition of tables and removal of update anomalies

Salesperson

sales-name	addr	dept	job-level

Sales-vacations

job-level	vacation-days

BCNF

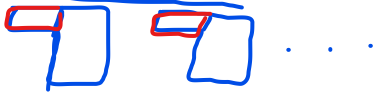
Logical Model vs. Physical Model

- Database tool vendors
  - Use the term *logical model* to refer to the conceptual data model
  - Use the term *physical model* to refer to the DBMS-specific implementation model (e.g., SQL tables)

Physical design

B+, 해시, 비트맵 (P.K 기본생성)　　분기, 월별, 연별 (범위 / 해시 / 조합　~ | ○○○

- Involves the selection of indexes (access methods), partitioning, and clustering of data



- The purpose of physical design is to optimize performance as closely as possible
- Denormalization 삭작추가
  - As part of the physical design, the global schema can sometimes be 조정 refined in limited ways to reflect processing (query and transaction) requirements if there are obvious, large gains to be made in efficiency
  - It consists of selecting dominant processes on the basis of high frequency, high volume, or explicit priority
  - Defining simple extensions to tables that will improve query performance
  - Evaluating total cost for query, update, and storage;
  - Considering the side effects, such as possible loss of integrity
- Partitioning
- Materialized views DDL crea~~te VI~~ 방법 정의 (Data 사전) ⟶ 실자료 저장

Test or 실제 Data로 실행

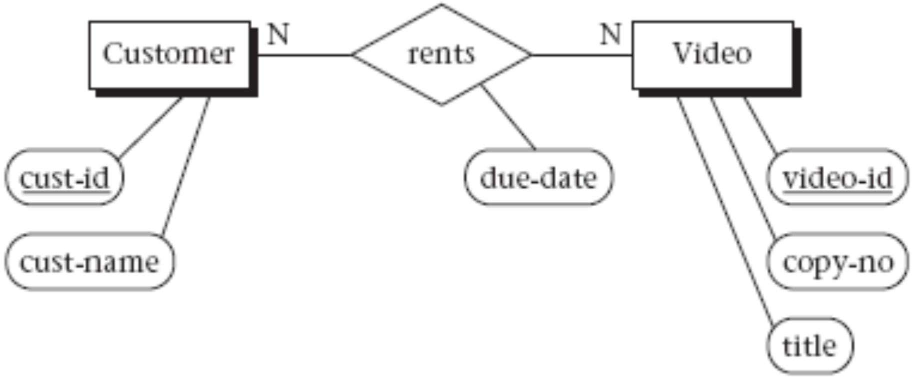
Database implementation, monitoring, and modification

- Once the design is completed, the database can be created through data definition language (DDL) of a DBMS
  - as well as to set up indexes and establish constraints, such as referential integrity
- DML
- As the database begins
  - Operation, monitoring indicates whether performance requirements are being met
  - If they are not being satisfied, modifications should be made to improve performance
- Other modifications may be necessary
  - When requirements change or when the end users' expectations increase with good performance
- Thus, the life cycle continues with monitoring, redesign, and modifications

관계형 = SQL → 이미 구조화, 웬만하면 변경X

Conceptual Data Modeling

- Entity-Relationship (ER)
  - First presented in 1976 by Peter Chen
  - Uses rectangles to specify entities
  - Uses diamond-shaped objects to represent the various types of relationships



- Unified Modeling Language (UML)
  - Introduced in 1997 by Grady Booch and James Rumbaugh
  - Has become a standard graphical language for specifying and documenting large-scale software systems
  - The data modeling component of UML (now UML-2) has a great deal of similarity with the ER model
- We will use both the ER model and UML to illustrate the data modeling and logical database design examples throughout this book

Conceptual Data Modeling

- The overriding emphasis is on simplicity and readability
- The goal of conceptual schema design
  - To capture real-world data requirements in a simple and meaningful way → understandable by both the database designer and the end user
    - The end user is the person responsible for accessing the database and executing queries and updates through the use of DBMS software
- ER model has two levels of definition
  - One that is quite simple
    - The simple level is the one used by most current design tools. It is quite helpful to the database designer who must communicate with end users about their data requirements
    - You simply describe, in diagram form, the entities, attributes, and relationships that occur in the system to be conceptualized, using semantics that are definable in a data dictionary
    - It is easy to learn and applicable to a wide variety of design problems
  - Another that is considerably more complex
    - Includes concepts from the semantic models of artificial intelligence and from competing conceptual data models
    - Useful to the database application programmer, because certain integrity constraints defined in the ER model relate directly to code—code that checks range limits on data values and null values, for example