Bazy danych. Seminarium II.

M.Tamer Ozsu, Patrick Valduriez,
Principles of Distributed Database Systems

rozdziały 1-4

Plan

1. DDBS – wprowadzenie.
2. Relacyjne BD – przypomnienie.
3. Sieci komputerowe – przypomnienie.
4. Architektura DDBMS.

DDBS

- DBMS - Database Management System
- DDBS – Distributed Database System

DDBS - oczekiwania

- Przezroczystość, *transparency*
  - Niezależność danych: odporność na zmianę organizacji danych
  - Przezroczystość położenia
  - Przezroczystość migracji
  - Przezroczystość istnienia kopii
  - Przezroczystość fragmentacji: podział relacji na zbiory pod-relacji
- Niezawodność, *reliability*: rozproszone transakcje
- Ulepszone wykonywanie operacji: szybciej
- Łatwiejsza rozbudowa systemu: bez ograniczeń

DDBS - wady

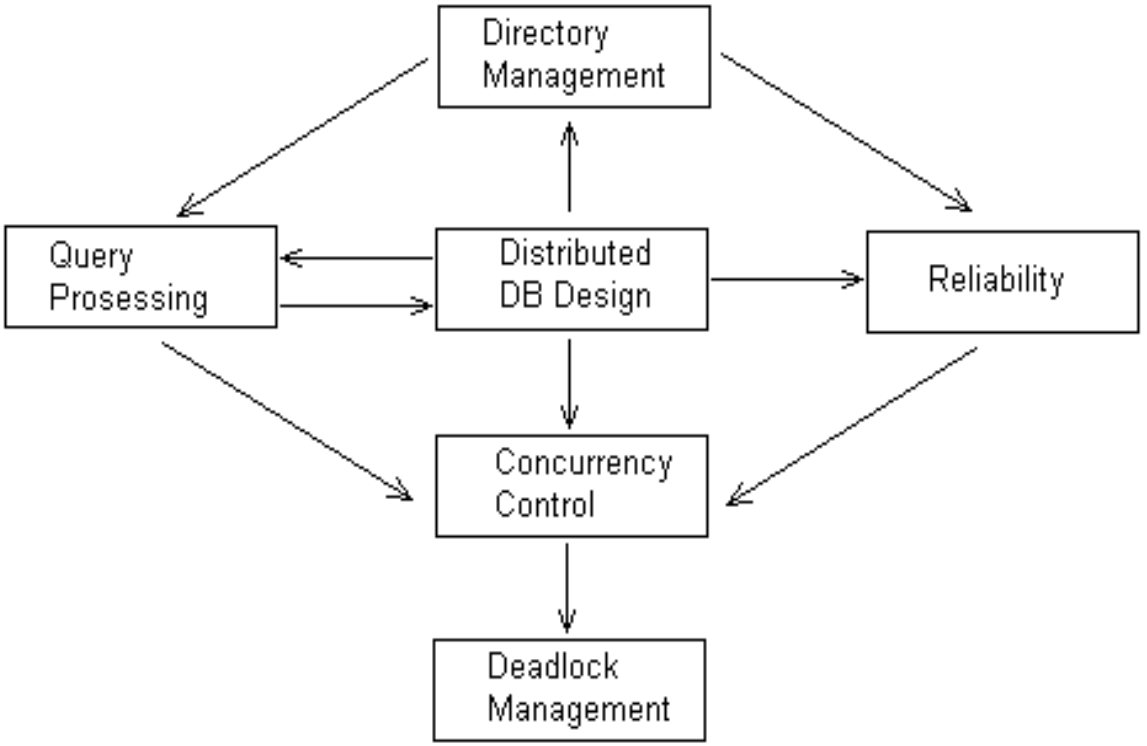
- Duży stopień skomplikowania systemu
- Duży koszt: np. potrzebny dodatkowy hardware
- Bezpieczeństwo

Budujemy DDBS...

1. Sposób przechowywania danych: *partitioned / replicated*.
2. Algorytmy do rozproszonych zapytań.
3. Mechanizmy zachowania spójności przy dostępie współbieżnym do bazy danych.
4. Zapobieganie zakleszczeniom.
5. Zapewnienie niezawodności.
7. Współpraca z systemem operacyjnym.

Rozwiązania tych problemów są ze sobą powiązane!

Budujemy DDBS...



Relacyjne bazy danych

Baza danych: zbiór danych o pewnej strukturze odzwierciedlający fragment rzeczywistości

Relacyjna baza danych: strukturę da się przedstawić jako relacje

Klucz: najmniejszy niepusty podzbiór atrybutów taki, że określają one jednoznacznia każdy wiersz w relacji

Normalizacje

Normalizacja: zastąpienie pewnego zbioru relacji innym zbiorem relacji o prostszej i bardziej regularnej strukturze (proces odwracalny)

Anomalie: redundancja, anomalie przy „update”, „insert”, „delete”

Postacie normalne:
1NF, 2NF, 3NF, BCNF, 4NF, 5NF

Działania na relacjach

Podstawowe:

- **Selekcja**: wybór pewnych wierszy.
- **Projekcja**: wybór pewnych kolumn.
- **Suma**
- **Różnica**
- **Produkt kartezjański**

Działania na relacjach

Dodatkowe:

- Złączenie naturalne
- Złączenie ogólne
- Zmiana nazwy kolumny

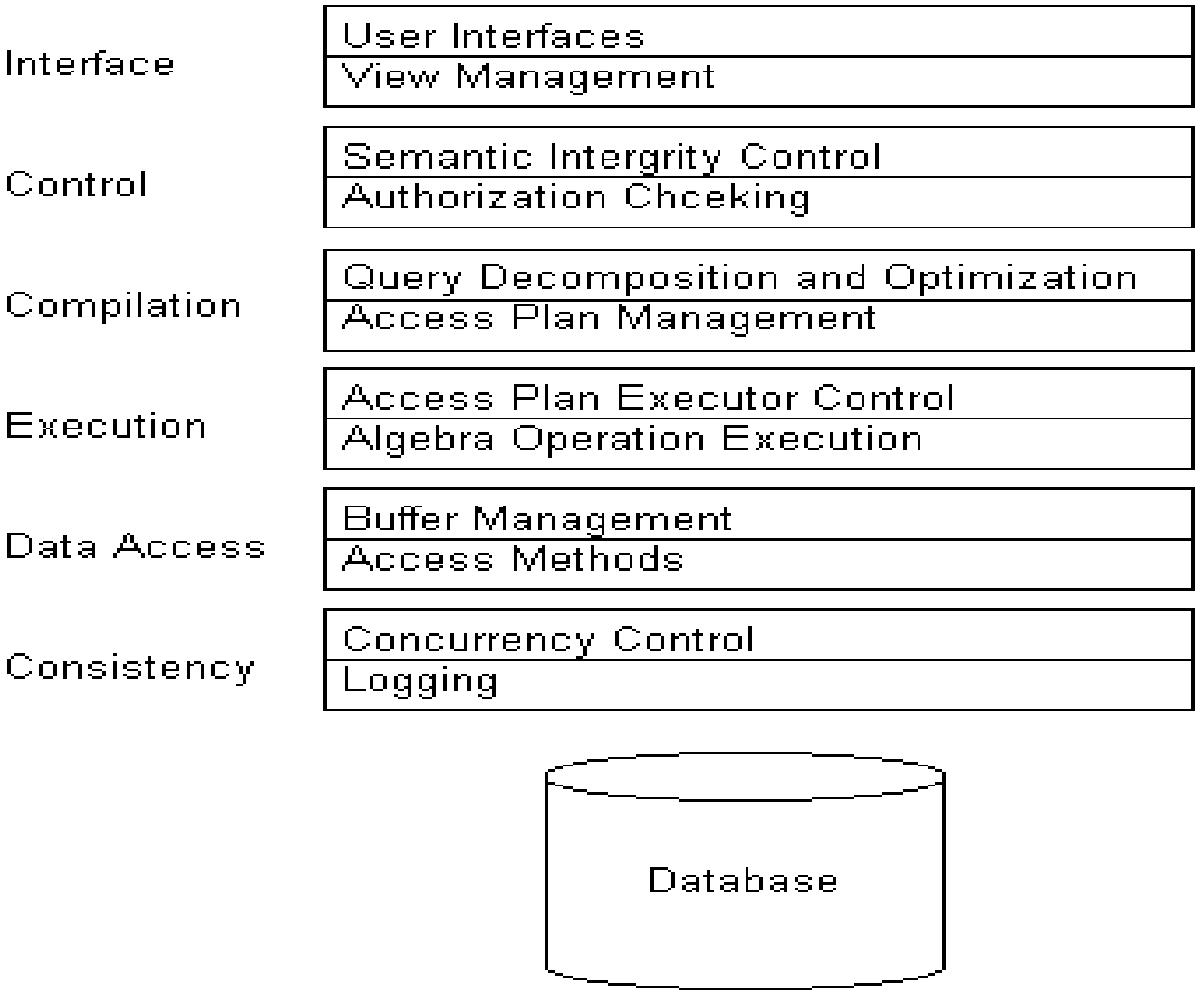
Języki zapytań

```
SELECT ename
FROM emp, dept
WHERE emp.deptno = dept.deptno
AND loc = 'CHICAGO';
```

równoważnie

$$\Pi_{ename}((\sigma_{loc=CHICAGO}\ \text{dept}) \bowtie_{deptno}\ \text{emp})$$

Relacyjne BD - warstwy



Sieci komputerowe

Różne charakterystyki transmisji:

- Przepustowość łącza
- Tryb przesyłania: *simplex, halfduplex, fullduplex*
- Dane przesyłane zwykle w ramkach:

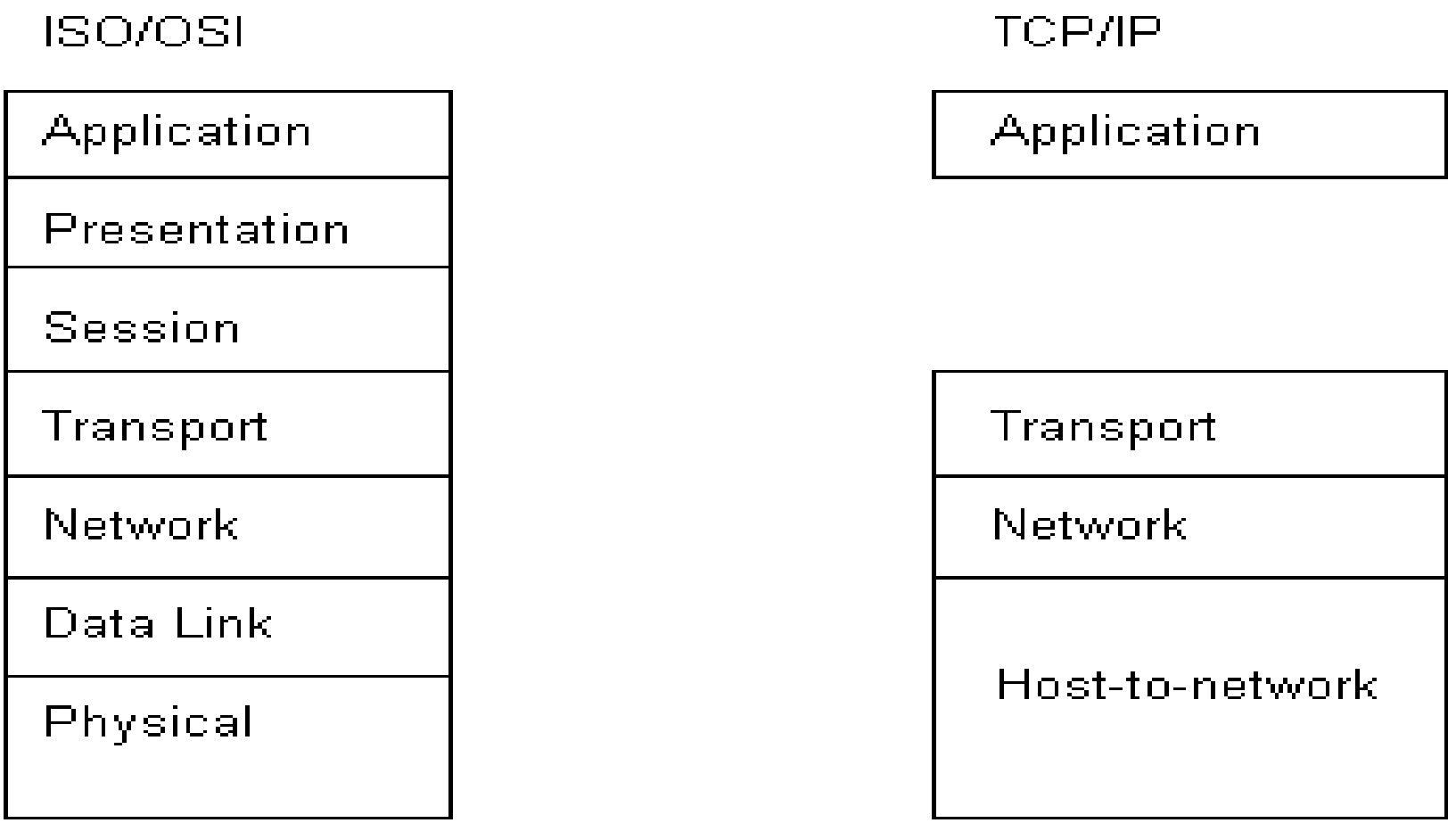
Klasyfikacja sieci komputerowych

Topologie:
gwiazda, pierścień, szyna, pełne, inne

Sposób transmisji:
point-to-point (*unicast*), multi-point (*broadcast*)

Wielkość:
WAN, MAN, LAN

Model warstwowy sieci



Architektura DDBMS-ów

Lata 70. - powstaje architektura ANSI/SPARC.

ANSI: American National Standards Institute
SPARC: Standards Planning and Requirements Committee

Architektura ANSI/SPARC

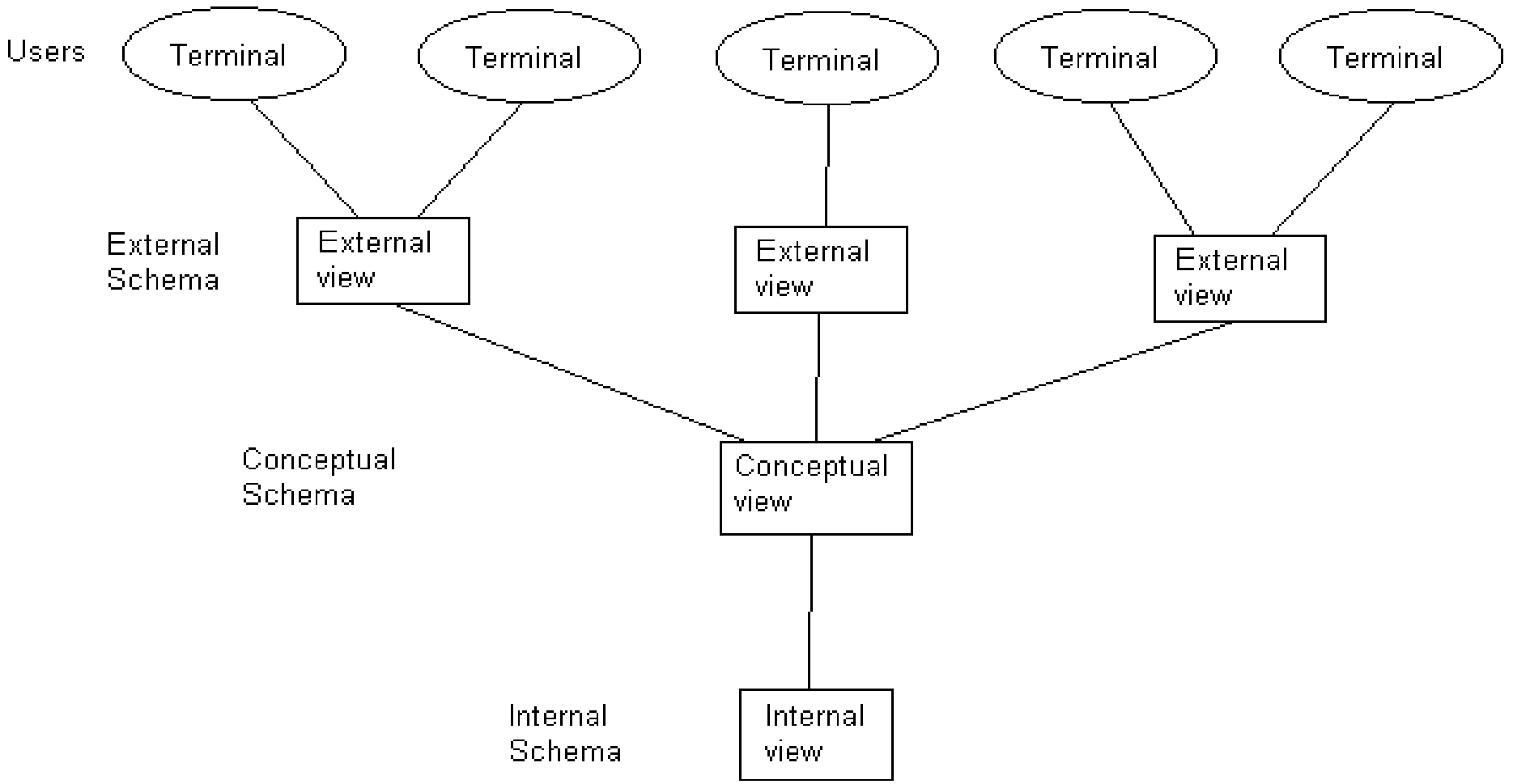
Jeden ze standardów organizowania i przechowywania danych. Trzy punkt widzenia:

External view: punkt widzenia użytkownika, np. programisty bazy danych

Conceptual view: punkt widzenia systemu bazy danych, abstrakcyjny opis kawałka rzeczywistości

Internal view: punkt widzenia rozmieszczenia danych.

Architektura ANSI/SPARC



Conceptual level, przykład

```
RELATON EMP [
    KEY = {ENO}
    ATTRIBUTES = {
        ENO    : CHARACTER(9)
        ENAME      : CHARACTER(15)
        TITLE      : CHARACTER(10)
    }   ]

RELATON PAY [
    KEY = {TITLE}
    ATTRIBUTES = {
        TITLE      : CHARACTER(10)
        SAL        : NUMERIC(6)
    }   ]

RELATON PROJ [
    KEY = {PNO}
    ATTRIBUTES = {
        PNO    : CHARACTER(7)
        PNAME      : CHARACTER(20)
        BUDGET : NUMERIC(7)
    }   ]
```

Internal level, przykład

```
INTERNAL REL EMPL [
   INDEX ON E# CALL EMINX
   FIELD = {
      HEADER: BYTE(1)
      E#    : BYTE(9)
      E:NAME: BYTE(15)
      TIT   : BYTE(10)
   }
]
```

External level, przykład

```
CREATE VIEW PAYROLL (ENO, ENAME, SAL)
AS
   SELECT EMP.ENO, EMP.ENAME, PAY.SAL
   FROM EMP, PAY
   WHERE EMP.TITLE = PAY.TITLE

CREATE VIEW BUDGET (PNAME, BUD)
AS
   SELECT PNAME, BUDGET
   FROM PROJ
```

Modele architektury dla DDBMS-ów

Kryteria:

Autonomiczność, *autonomy*

Stopień rozproszenia, *distribution*

Niejednorodność, *heterogeneity*

Autonomiczność

Jest to stopień, w jakim lokalna baza danych może działać niezależnie od innych baz:

im wyższa autonomiczność, tym większa redundancja (-> centralizowanie)

im niższa autonomiczność, tym większa zawodność i mniejsza elastyczność

Autonomiczność

Autonomiczność projektu: lokalny SZBD może używać takiego modelu danych jakiego chce i takiego zarządzania transakcjami jakiego chce.

Autonomiczność komunikacji: lokalny SZBD może decydować jaki typ informacji będzie dostarczał do innych SZBD-ów.

Autonomiczność wykonania: lokalny SZBD wykonuje kierowane do niego transakcji w sposób wygodny dla niego.

Autonomiczność

Pół-autonomiczność, *semiautonomy*: system złożony z SZBD-ów działających niezależnie, ale zrzeszonych w „federację” (udostępniają innym SZBD swoje lokalne dane)

Całkowita niezależność, *total isolation*: system składa się z izolowanych SZBD, żaden SZBD nie wie o istnieniu innych SZBD – brak globalnej kontroli nad wykonywaniem zapytań

Sposób rozproszenia

Fizyczne rozproszenie danych pomiędzy komputery w sieci:

client/server: różne funkcje

peer-to-peer: każdy lokalny SZBD ma taką samą funkcję jak inne

Niejednorodność

Są to ukryte różnice pomiędzy różnymi SZBD:

- architektury węzłów,
- środki transmisji,
- protokoły komunikacyjne,
- różne modele danych (te same języki zapytań),
- różne języki zapytań (te same modele danych).

Jednorodność, *homogeneity*

Architektura DDBMS

A = autonomy (autonomiczność)
D = distribution (rozproszenie)
H = heterogeneity (niejednorodność)

A0 – silne związanie, A1 – półautonomicznośc, A2 – całkowita izolacja
D0 – bez rozproszenia, D1 – client/server, D2 – peer-to-peer
H0 – jednorodność, H1 – niejednorodność

Możemy rozważać systemy (Ax, Dy, Hz), np.
(A0, D0, H0) – wieloprocesory (współdzielone jest wszystko)
(A2, D2, H1) – 2 hosty z różnych podsieci

Architektura klient/serwer

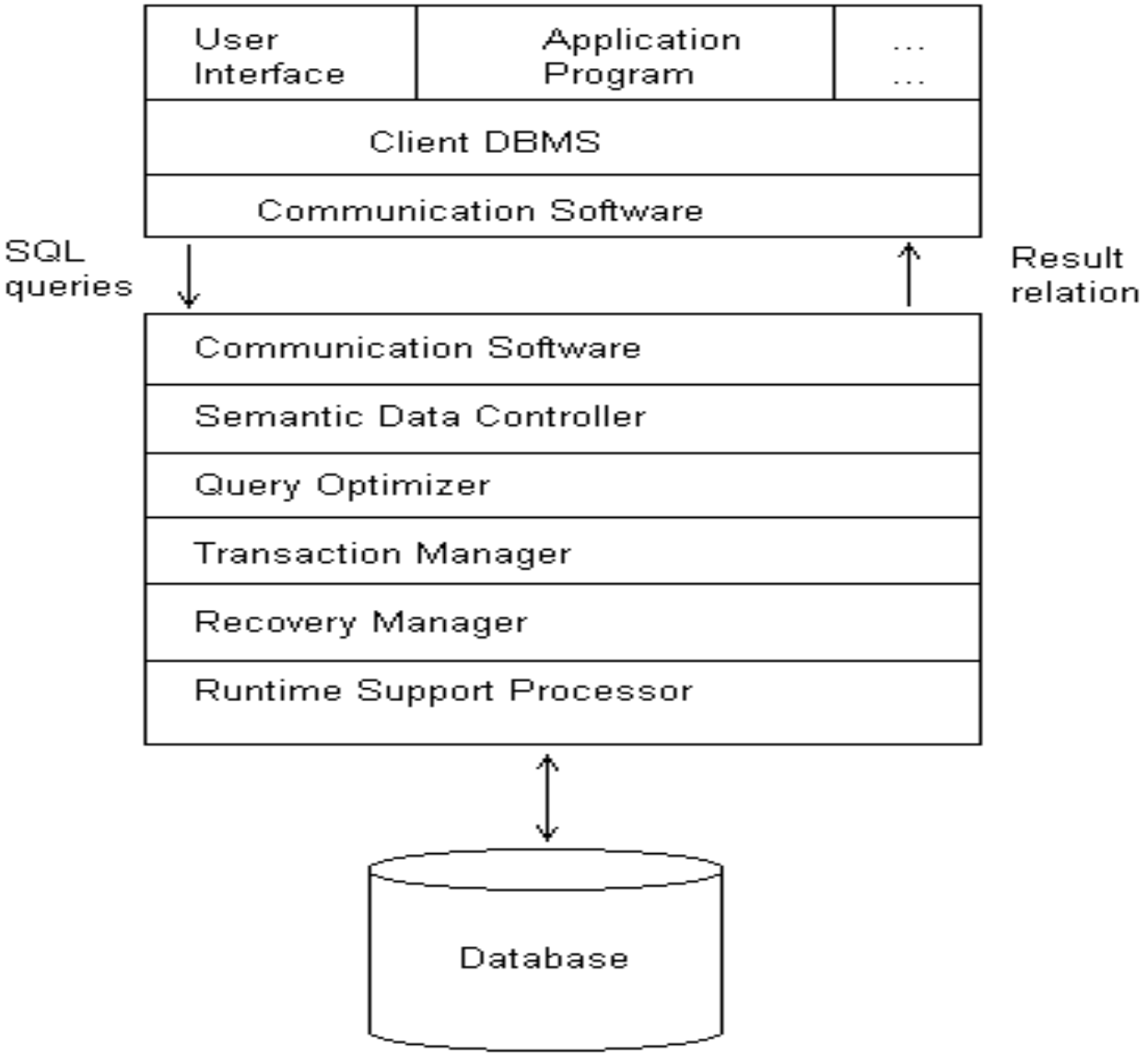
Zróżnicowanie funkcji.

Serwer: zarządzanie większością procesów (realizowanie zapytań, optymalizacja, transakcje, operacje na danych fizycznych).

Klient: tylko wysyłanie zapytań (nawet bez optymalizacji).

Multiple-client, single-server,
Multiple-client, multiple-server,

Architektura klient/serwer

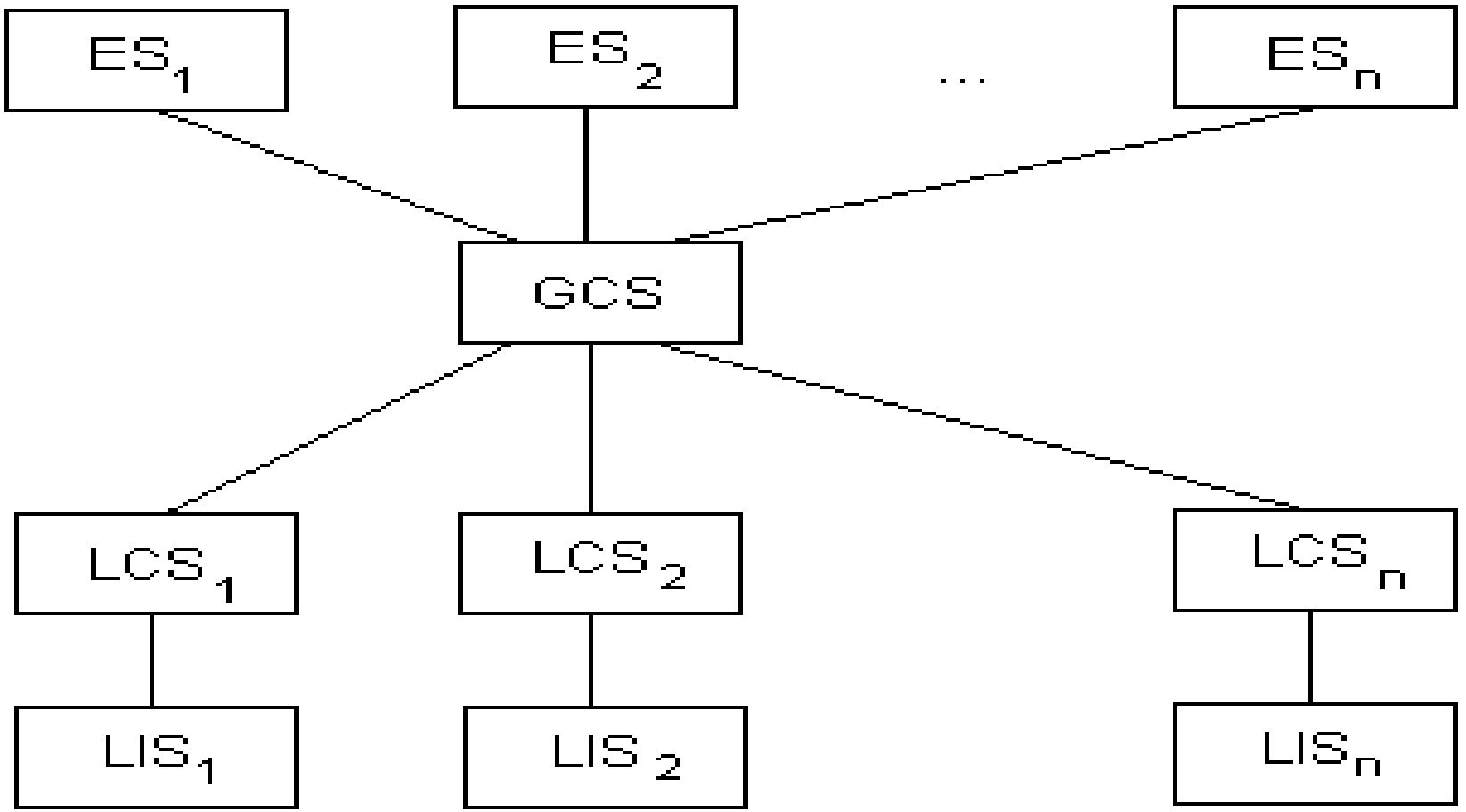


Architektura peer-to-peer

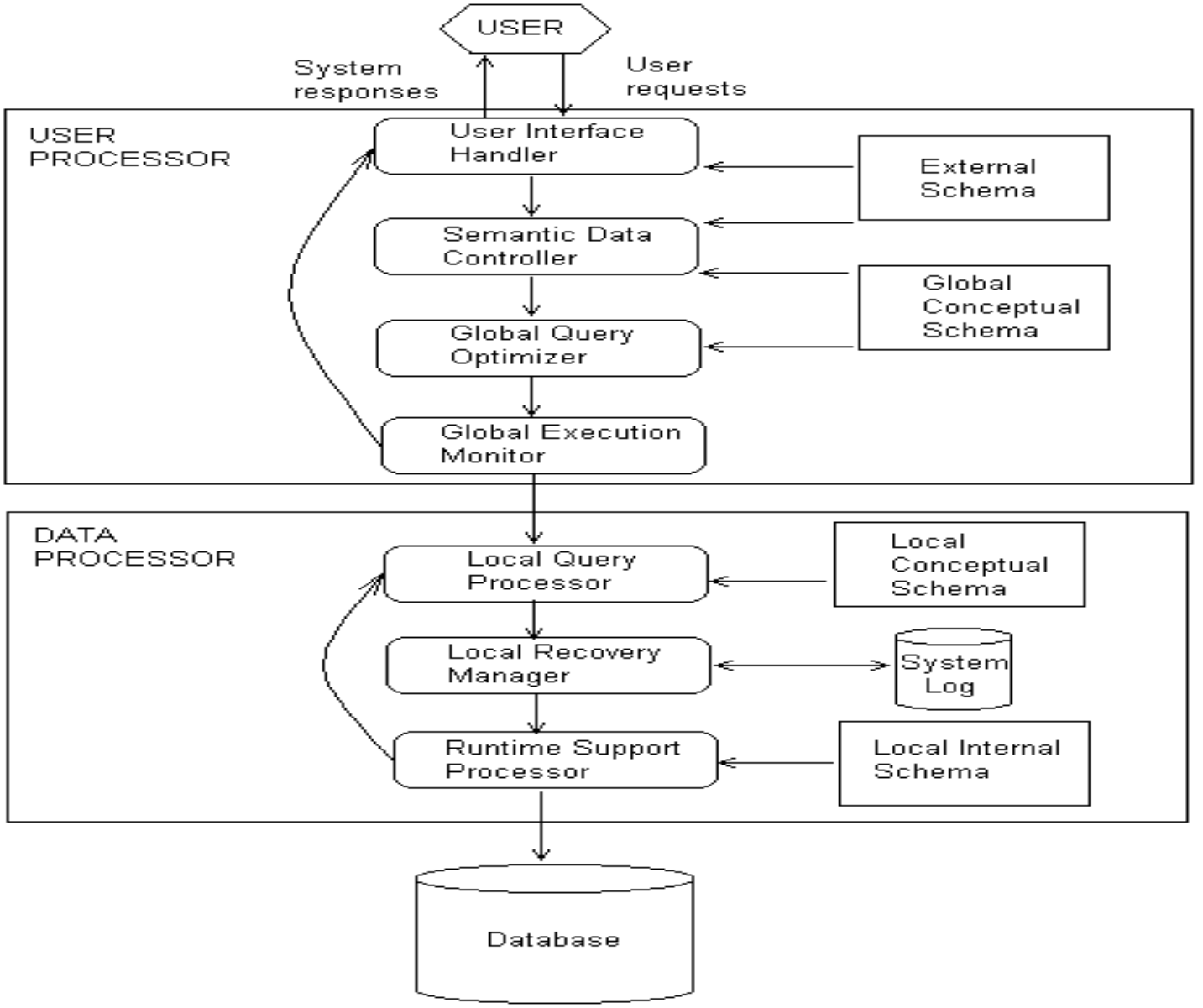
Fizyczna organizacja danych na każdej maszynie może być (i zwykle jest) inna => każdy lokalny SZBD musi mieć swój własny *internal view* /schema.

LIS: Local Internal Schema
LCS: Local Conceptual Schema
GCS: Global Conceptual Schema
ES: External Schema

Architektura peer-to-peer



Architektura DDBMS



Podsumowanie

Przedstawione rozwiązania są „modelowe”. Autorzy twierdzą, że są one dobre do pokazania idei DDBMS-ów, ale nie zawsze da się je zaimplementować.