Database Design

• Purpose: organize to achieve efficiency, ...
• Considerations
  – time/space tradeoff for application
  – balance application characteristics, management requirements, competing theories, ...
  – tool support and tool development based on theory
• Approach
  – systematic
  – transformational
  – guided by theory, but tempered by application semantics
  – adjusted by application-dependent cost analysis
Design Steps

ORM application model  \rightarrow  ORM hypergraph  \leftarrow  Model-theoretic view

Generation of basic constraints

ORM hypergraph  \leftrightarrow  DB schemes and constraints

Scheme generation

design transformations  \leftrightarrow  time/space adjustments

If transformations and adjustments preserve information and constraints, the generated schemes and constraints are equivalent to the originals in the model-theoretic view.

efficiently organized

not efficiently organized

DesignIntro: 2
Hypergraphs

- Graph = (V, E)
  - V = set of vertices
  - E = set of edges (two vertices; one for unordered loops)
    - e 0 E can be unordered: e = {x, y}
    - e 0 E can be ordered: e = (x, y)

- Hypergraph = (V, E)
  - V = set of vertices
  - E = set of edges (can have more than two vertices)
    - e 0 E can be unordered: e = {x_1, x_2, …, x_n}
    - e 0 E can be ordered: e = (x_1, …, x_m; y_1, …, y_k)
      = e: x_1, …, x_m 6 y_1, …, y_k = x_1, …, x_m 6 y_1, …, y_k
ORM Hypergraphs

• A regular hypergraph with:
  – object sets as vertices
  – relationship sets as edges

• Plus:
  – Generalization/Specialization
  – Optional-participation markers
  – General constraints

• Example:
Conversion to ORM Hypergraph

\[ \frac{1}{2} \forall x (H(x) \Rightarrow \exists y, z, w (D(y)H(x)I(z)J(w))) \]
Design Algorithm Essentials

1. Combine functional edges with the same tail object set(s) into a scheme. (The tail object set(s) and any others in a 1-1 correspondence are keys.)

   \[
   \begin{align*}
   &A \xrightarrow{} B \xleftarrow{} C \\
   &A \xrightarrow{} B \xrightarrow{} C \\
   \end{align*}
   \]

2. Each nonfunctional edge is a scheme. (All object sets constitute the key.)

   \[
   \begin{align*}
   &A \xrightarrow{} B \\
   &A \xrightarrow{} B \xrightarrow{} C \\
   \end{align*}
   \]

3. Each stand-alone object set is a scheme. (The lone object set is a key.)

   \[
   \begin{align*}
   &A \\
   &A \\
   \end{align*}
   \]
B&B Example – ORM Diagram

Guest with Name lives on StreetNr in City

a + b > 0

Guest has reservation for Room on ArrivalDate for NrDays

Name, StreetNr, City -> Guest

Room, ArrivalDate -> Guest, NrDays
B&B Example – ORM Hypergraph

```plaintext
Room

RoomNr

Name

Cost

StreetNr

City

a+b > 0

b:*a:1

Room

Guest

GuestNr

ArrivalDate

NrDays

NrBeds

Cost
```

DesignIntro: 8
B&B Example – Optionals Removal
B&B Example – Nonlexical Removal
B&B Example –
Generated Database Scheme

Room(RoomNr, RoomName, NrBeds, Cost)

Guest(GuestNr, GuestName, StreetNr, City)

Reservation(GuestNr, RoomNr, ArrivalDate, NrDays)

Room[RoomNr] g Reservation[RoomNr]
Guest[GuestNr] = Reservation[GuestNr]