Abstraction

- Concrete: directly executable/storable
- Abstract: not directly executable/storable
  - automatic translation (as good as executable/storable)
  - systematic translation
  - ad hoc translation
  - not translatable (incomplete or unclear)

The 3-Level Architecture

- View 1
- View 2
- View n

Abstract

- Conceptual Database
- Physical Database

Independence

Abstract Data Type (ADT)

- Abstract Description of a Data Type
- (Set of Values, Set of Operations)
- Examples:
  - Integer = \{0, -1, 1, -2, 2, ..., \}
  - Stack = \{(all possible stacks, (push, pop, top, empty))\}
  - Database = \{(all possible DB states, (insert, remove, retrieve))\}
- Description of all possible DB States:
  - Description of Conceptual Database
  - Schema, Schema, Model

Entity-Relationship (ER) Model

- Entity -- object
- Relationship -- connection among objects
- Attribute -- properties of objects and connections

ER Application Model

ISA – Generalization/Specialization

- ISA under construction
- Hotel
- Complex: Hotel
- Room
- Guest
- ...
Cardinality Relationships

1-1
m-1; 1-m seen the other way
m-n

Entity Key
Uniquely Identifies an Individual Entity

Keys: A or BC

Relationship Key
Uniquely Identifies an Individual Relationship

Key: A or B
Key: A
Key: AB

Key Attributes for Relationships
Relationship attributes are sometimes needed to uniquely identify an individual relationship.

Depending on the semantics, the key could be: ABC, AC, BC, or C.

Keys

- Superkey: A set of attributes that uniquely identifies
- Minimal Key: A minimal superkey.
- Candidate Key: Same as minimal key.
- Primary Key: A chosen candidate key.

Scheme Generation

- Generate a scheme for each entity set (except specializations)
  - The attributes are the attributes of the entity set.
  - The keys are the candidate keys of the entity set; choose a primary key.
- For each relationship set, adjust or add a scheme.
  - 1-1: merge schemes; add any relationship attributes.
  - 1-m, m-1: to the scheme on the many-side, add the primary-key attribute(s) of the one-side; add any relationship attributes.
  - m-m: make a new scheme from the primary-key attributes of the connected entity sets and any relationship attributes; the key is the composite of the primary-key attributes if there are no key relationship attributes—otherwise semantically determine the key.
- Generate a scheme for each specialization.
  - The attributes are the attributes of the specialization plus the primary key attribute(s) of the generalization.
  - The key is the primary key of the generalization.
Scheme Generation – Example

Generated schemes, with candidate keys underlined and primary keys double underlined:

- Room(RoomNr, Name, NrBeds, Cost)
- Guest(GuestNr, Name, StreetNr, City)
- Reservation(GuestNr, RoomNr, ArrivalDate, NrDays)
- RoomUnderConstruction(RoomNr, CompletionDate)

Sample Database Instance

<table>
<thead>
<tr>
<th>RoomNr</th>
<th>Name</th>
<th>NrBeds</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Kennedy</td>
<td>2</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>Nixon</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>3</td>
<td>Carter</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>4</td>
<td>Blue</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>Green</td>
<td>1</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GuestNr</th>
<th>Name</th>
<th>StreetNr</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Smith</td>
<td>12 Maple</td>
<td>Boston</td>
</tr>
<tr>
<td>102</td>
<td>Carter</td>
<td>10 Main</td>
<td>Hartford</td>
</tr>
<tr>
<td>103</td>
<td>Jones</td>
<td>6 Elm</td>
<td>Hartford</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reservation NrDays</th>
</tr>
</thead>
<tbody>
<tr>
<td>101 10 May 2</td>
</tr>
<tr>
<td>101 20 May 1</td>
</tr>
<tr>
<td>101 15 May 2</td>
</tr>
<tr>
<td>102 10 May 5</td>
</tr>
<tr>
<td>103 12 May 3</td>
</tr>
<tr>
<td>104 10 May 2</td>
</tr>
<tr>
<td>104 17 May 2</td>
</tr>
<tr>
<td>104 24 May 2</td>
</tr>
<tr>
<td>105 15 May 7</td>
</tr>
<tr>
<td>106 11 May 2</td>
</tr>
</tbody>
</table>

Sample Database Instance

- Room(RoomNr integer primary key, Name char(20) unique, NrBeds integer, Cost float)
- Guest(GuestNr integer primary key, Name char(20), StreetNr integer, City char(15), unique (Name, StreetNr, City))
- Reservation(GuestNr integer references Guest, RoomNr integer references Room, ArrivalDate char(6), NrDays integer, primary key (RoomNr, ArrivalDate))

Relational Algebra

- An Algebra is a Pair: (set of values, set of operations)
- Note that an Algebra is the same idea as an ADT
- Relational Algebra: (relations, relational operators)
  - set of values = relations
  - set of operations = relational operators
- Relational Operators
  - update operators:
    - insert a tuple
    - delete one or more tuples
    - modify one or more tuples
  - retrieval operators: {σ, π, , −, ⋈, ρ, ·, |−|}

σ – Selection

General Form: σ<condition> <relation>

Examples:

- \( σ_{\text{Cost} > 75} r \)
- \( σ_{\text{ArrivalDate} = \text{10 May}} s \)
- \( σ_{\text{ArrivalDate} = \text{10 May} \land \text{NrDays} = 2} s \)

π – Projection

General Form: π<attributes> <relation>

Examples:

- \( π_{\text{City}} g \)
- \( π_{\text{GuestNr}, \text{RoomNr}} s \)
Closed Set of Operators

- Results are relations
- Closed implies we can nest operations in expressions
- Example:

\[ \pi_{\text{Guest Nr}, \text{Room Nr}} (\text{Arrival Date} = 10 \text{ May}) \]

<table>
<thead>
<tr>
<th>Guest Nr</th>
<th>Room Nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>102</td>
<td>3</td>
</tr>
<tr>
<td>104</td>
<td>4</td>
</tr>
</tbody>
</table>

Set Operators: \( \cap, - , \cup \)

- Example:

\[ \pi_{\text{Name}} (\text{Name} \cap \text{Name} = \text{Name}) \]

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carter</td>
</tr>
<tr>
<td>Green</td>
</tr>
</tbody>
</table>

\[ \pi_{\text{Room Nr}} (\text{Cost} < 75) \]

<table>
<thead>
<tr>
<th>Room Nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

\[ \pi_{\text{Room Nr}} (\text{Arrival Date} = 10 \text{ May}) \]

<table>
<thead>
<tr>
<th>Room Nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

Note: schemes must be compatible.

\( \rho \) – Renaming

General Form: \( \rho_{\text{old attribute}} \rightarrow \text{new attribute} \) \( \pi_{\text{relation}} \)

Examples:

\[ \pi_{\text{Room Nr}, \text{Name}, \text{Nr Beds}, \text{Cost}, \text{Guest Nr}, \text{Street Nr}, \text{City}} \]

<table>
<thead>
<tr>
<th>Nr Beds</th>
<th>Room Nr</th>
<th>Cost</th>
<th>Guest Nr</th>
<th>Street Nr</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>100</td>
<td>101</td>
<td>Hartford</td>
<td>Boston</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>200</td>
<td>102</td>
<td>Hartford</td>
<td>Boston</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>150</td>
<td>103</td>
<td>Boston</td>
<td></td>
</tr>
</tbody>
</table>

Note: The old attribute must be in the scheme and the new attribute must not be in the scheme.

\( \times \) – Cross Product

Examples:

\[ \pi_{\text{Nr Beds}} (\text{Nr Beds} \times \text{Room Nr}) \]

<table>
<thead>
<tr>
<th>Nr Beds</th>
<th>Room Nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Note: The intersection of the schemes must be empty.

\( | \times | \) – Natural Join

Examples:

\[ \pi_{\text{Room Nr}, \text{Name}, \text{Nr Beds}, \text{Cost}, \text{Guest Nr}, \text{Street Nr}, \text{City}} \]

<table>
<thead>
<tr>
<th>Room Nr</th>
<th>Name</th>
<th>Nr Beds</th>
<th>Cost</th>
<th>Guest Nr</th>
<th>Street Nr</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Carter</td>
<td>2</td>
<td>80</td>
<td>102</td>
<td>10 Main</td>
<td>Hartford</td>
</tr>
<tr>
<td>5</td>
<td>Green</td>
<td>1</td>
<td>50</td>
<td>105</td>
<td>10 Main</td>
<td>Boston</td>
</tr>
</tbody>
</table>

\[ \pi_{\text{Name}} (\text{Name} = \text{Name}) \]

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carter</td>
</tr>
<tr>
<td>Green</td>
</tr>
</tbody>
</table>

\[ \pi_{\text{Room Nr}} (\text{Cost} < 75) \]

<table>
<thead>
<tr>
<th>Room Nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

\[ \pi_{\text{Room Nr}} (\text{Arrival Date} = 10 \text{ May}) \]

<table>
<thead>
<tr>
<th>Room Nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

Natural Join – Examples

<table>
<thead>
<tr>
<th>A B</th>
<th>C D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>1 3</td>
</tr>
<tr>
<td>3 4</td>
<td>1 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A B C</th>
<th>D E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
<td>1 2</td>
</tr>
<tr>
<td>2 3 4</td>
<td>1 2</td>
</tr>
<tr>
<td>3 4 5</td>
<td>1 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A B C</th>
<th>D E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
<td>1 2</td>
</tr>
<tr>
<td>2 3 4</td>
<td>1 2</td>
</tr>
<tr>
<td>3 4 5</td>
<td>1 2</td>
</tr>
</tbody>
</table>
Query Examples

List names and cities of guests arriving on 15 May.
\[ \pi_{\text{Name,City}}(\text{ArrivalDate} = 15 \text{ May}) \]

List names of each guest who has a reservation for a room that has the same name as the guest’s name.
\[ \pi_{\text{Name}}(g \mid s \}_r \]

List names of guests who have a reservation for rooms with two beds.
\[ \pi_{\text{Name}}(g \mid s \}_r (\text{RoomNr} \in 2) \]

More Query Examples

List the names of guests from Hartford who are arriving after 10 May.
\[ \pi_{\text{Name}}(\text{GuestNr,Name,City = Hartford} \mid \text{ArrivalDate} > 10 \text{ May}) \]

List names of rooms for which no guest is arriving on 10 May.
\[ \pi_{\text{Name}}(r \mid \text{RoomNr} \in (\text{RoomNr} \text{ArrivalDate = 10 May})) \]

SQL

Correspondence with Relational Algebra
Assume \( r(\text{AB}) \) and \( s(\text{BC}) \).

Select \( A \) from \( r \)
where \( B = 1 \)

Select \( B \) from \( r \)
minus
Select \( B \) from \( s \)

Select \( A \) “D” from \( r \)
where \( r.B = s.B \)

Select \( A, B, C \) from \( r, s \)
where \( r.B = s.B \)

SQL: Basic Retrieval

Get full details of rooms.

Select \( \text{Name, NrBeds, Cost} \)
from \( \text{Room} \)

Get costs.

Select \( \text{Cost} \)
from \( \text{Room} \)
Select \( \text{Cost} \)
from \( \text{Room} \)
Select distinct \( \text{Cost} \)
from \( \text{Room} \)

SQL: Aggregate Operations

Get average cost.

Select \( \text{avg(Cost)} \)
from \( \text{Room} \)

Select \( \text{avg(Cost)} \) “AVES”
from \( \text{Room} \)

Also: min, max, sum, count.

SQL: Sorting

Get distinct costs in descending order.

Select distinct \( \text{Cost} \)
from \( \text{Room} \)
order by \( \text{Cost} \) desc

Get cost and number of beds, sorted with beds in ascending order and cost in descending order.

Select distinct \( \text{NrBeds, Cost} \)
from \( \text{Room} \)
order by \( \text{NrBeds asc, Cost desc} \)

<table>
<thead>
<tr>
<th>NRBEDS</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
</tr>
</tbody>
</table>
SQL: Grouping
Get the average cost of rooms with the same number of beds.

```
select NrBeds, avg(Cost)
from Room
group by NrBeds
```

Get guest numbers of guests who have reservations for a total of 5 or more days.

```
select GuestNr
from Reservation
group by GuestNr
having sum(NrDays) >= 5
```

---

SQL: Nested Queries
Get Names and Addresses of guests who have reservations for a total of 5 or more days.

```
select Name, StreetNr, City
from Guest
where GuestNr in (
  select GuestNr
  from Reservation
  group by GuestNr
  having sum(NrDays) >= 5)
```

---

SQL: Multiple-Relation Retrieval
Get names of guests whose names match room names.

```
select g.Name
from Guest g, Room r
where g.Name = r.Name
and g.GuestNr = s.GuestNr
and s.RoomNr = r.RoomNr
```

---

SQL: Select/Project/Join Queries
Get names and addresses of guests arriving on 15 May.

```
select Name, StreetNr, City
from Guest g, Reservation s
where g.GuestNr = s.GuestNr
and ArrivalDate = "15 May"
```

---

SQL: Negation
Get names and addresses of guests not arriving on 15 May.

```
select Name, StreetNr, City
from Guest
minus
select Name, StreetNr, City
from Guest g, Reservation s
where g.GuestNr = s.GuestNr
and ArrivalDate = "15 May"
```

---

SQL: Joining a Relation with Itself
Get pairs of guest numbers of guests arriving on the same date. (The pairs should be unique.)

```
select s.GuestNr, t.GuestNr
from Reservation s, Reservation t
where s.ArrivalDate = t.ArrivalDate
and s.GuestNr < t.GuestNr
```

---