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
- Conceptual Database
- Physical Database
- Abstract
- Concrete
- Implementation 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
  - Scheme, Schema, Model

Entity-Relationship (ER) Model

- Entity -- object
- Relationship -- connection among objects
- Attribute -- properties of objects and connections
Chapter 2 - 5

ER Application Model

Chapter 2 - 6

ISA – Generalization/Specialization
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
Chapter 2 - 9

**Relationship Key**
Uniquely Identifies an Individual Relationship

- ![Diagram](image1.png)
  - Key: A or B
- ![Diagram](image2.png)
  - Key: A
- ![Diagram](image3.png)
  - Key: AB

Chapter 2 - 10

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

- ![Diagram](image4.png)

  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)

SQL DDL

create table Room(
    RoomNr integer primary key,
    Name char(20) unique,
    NrBeds integer,
    Cost float
);

create table Guest(
    GuestNr integer primary key,
    Name char(20),
    StreetNr char(20),
    City char(15),
    unique (Name, StreetNr, City)
);

create table Reservation(
    GuestNr integer references Guest,
    RoomNr integer references Room,
    ArrivalDate char(6),
    NrDays integer,
    primary key (RoomNr, ArrivalDate)
);
Sample Database Instance

\[
\begin{array}{cccc}
   r & = & \text{Room}(\text{RoomNr} , \text{Name} , \text{NrBeds} , \text{Cost}) \\
   g & = & \text{Guest}(\text{GuestNr} , \text{Name} , \text{StreetNr} , \text{City}) \\
\end{array}
\]

\[
\begin{array}{cccc}
   1 & \text{Kennedy} & 2 & 90 \\
   2 & \text{Nixon} & 2 & 80 \\
   3 & \text{Carter} & 2 & 80 \\
   4 & \text{Blue} & 1 & 60 \\
   5 & \text{Green} & 1 & 50 \\
   101 & \text{Smith} & 12 \text{Maple} & \text{Boston} \\
   102 & \text{Carter} & 10 \text{Main} & \text{Hartford} \\
   103 & \text{Jones} & 6 \text{Elm} & \text{Hartford} \\
   104 & \text{Smith} & 4 \text{Oak} & \text{Providence} \\
   105 & \text{Green} & 10 \text{Main} & \text{Boston} \\
   106 & \text{Johnson} & 15 \text{Main} & \text{Boston} \\
\end{array}
\]

\[
\begin{array}{cccc}
   s & = & \text{Reservation}(\text{GuestNr} , \text{RoomNr} , \text{ArrivalDate} , \text{NrDays}) \\
\end{array}
\]

\[
\begin{array}{cccc}
   101 & 1 & 10 \text{May} & 2 \\
   101 & 2 & 20 \text{May} & 1 \\
   101 & 3 & 15 \text{May} & 2 \\
   102 & 3 & 10 \text{May} & 5 \\
   103 & 1 & 12 \text{May} & 3 \\
   104 & 4 & 10 \text{May} & 2 \\
   104 & 4 & 17 \text{May} & 2 \\
   104 & 4 & 24 \text{May} & 2 \\
   105 & 1 & 15 \text{May} & 7 \\
   106 & 2 & 11 \text{May} & 2 \\
\end{array}
\]

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: \( \{ \sigma, \pi, \cup, -, \cap, \rho, \times, |X| \} \)
σ – Selection

General Form: $\sigma_{\text{condition}} <\text{relation}>$

Examples:

$\sigma_{\text{Cost} > 75} \quad r$

<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>
</tbody>
</table>

$\sigma_{\text{ArrivalDate} = 10 \text{ May} \land \text{NrDays} > 2} \quad S$

<table>
<thead>
<tr>
<th>GuestNr</th>
<th>RoomNr</th>
<th>ArrivalDate</th>
<th>NrDays</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>3</td>
<td>10 May</td>
<td>5</td>
</tr>
</tbody>
</table>

π – Projection

General Form: $\pi_{\text{attributes}} <\text{relation}>$

Examples:

$\pi_{\text{City}} \quad g$

<table>
<thead>
<tr>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston</td>
</tr>
<tr>
<td>Hartford</td>
</tr>
<tr>
<td>Providence</td>
</tr>
</tbody>
</table>

$\pi_{\text{GuestNr}, \text{RoomNr}} \quad s$

<table>
<thead>
<tr>
<th>GuestNr</th>
<th>RoomNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>3</td>
</tr>
<tr>
<td>102</td>
<td>3</td>
</tr>
<tr>
<td>103</td>
<td>1</td>
</tr>
<tr>
<td>104</td>
<td>4</td>
</tr>
<tr>
<td>105</td>
<td>1</td>
</tr>
<tr>
<td>106</td>
<td>2</td>
</tr>
</tbody>
</table>
Closed Set of Operators

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

\[ \Pi_{\text{GuestNr}, \text{RoomNr}} \sigma_{\text{ArrivalDate} = 10 \text{ May}} S \]

<table>
<thead>
<tr>
<th>GuestNr</th>
<th>RoomNr</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: \( \cap, -, \cup \)

- \( \Pi_{\text{Name}} r \cap \Pi_{\text{Name}} g \)
  
<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carter</td>
</tr>
<tr>
<td>Green</td>
</tr>
</tbody>
</table>

- \( \Pi_{\text{RoomNr}} r - \Pi_{\text{RoomNr}} s \)
  
<table>
<thead>
<tr>
<th>RoomNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

- \( \Pi_{\text{RoomNr}} \sigma_{\text{Cost} < 75} r \cup \Pi_{\text{RoomNr}} \sigma_{\text{ArrivalDate} = 10 \text{ May}} s \)
  
<table>
<thead>
<tr>
<th>RoomNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Note: schemes must be compatible.
\( \rho \) – Renaming

General Form: \( \rho \ <\text{old attribute}> - <\text{new attribute}> <\text{relation}> \)

Examples:

\[
\pi_{\text{RoomName}} \rho_{\text{Name}} - \text{RoomName} \sigma_{\text{Cost} < 75} r \\
\]

\[
\pi_{\text{Nr}, \text{Name}} \rho_{\text{RoomNr} - \text{Nr}} r \cap \pi_{\text{Nr}, \text{Name}} \rho_{\text{GuestNr} - \text{Nr}} g \\
\]

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

\( \times \) – Cross Product

\[
\pi_{\text{NrBeds}} r \times \pi_{\text{RoomNr}} S \\
\]

<table>
<thead>
<tr>
<th>NrBeds</th>
<th>RoomNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NrBeds</th>
<th>RoomNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Note: The intersection of the schemes must be empty.
\( \times \) – Natural Join

\[
 r \times g = \\
\Pi_{\text{RoomNr, Name, NrBeds, Cost, GuestNr, StreetNr, City}} \sigma_{\text{Name} = \text{Name}'} (r \times \rho_{\text{Name} = \text{Name}'} g)
\]

<table>
<thead>
<tr>
<th>RoomNr</th>
<th>Name</th>
<th>NrBeds</th>
<th>Cost</th>
<th>GuestNr</th>
<th>StreetNr</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>

Natural Join – Examples

\[
\begin{array}{ccc}
A & B & \times & B & C \\
\hline
1 & 2 & & 1 & 2 \\
3 & 2 & & 2 & 3 \\
4 & 5 & & 5 & 7 \\
6 & 7 & & 5 & 8 \\
\end{array} = \\
\begin{array}{ccc}
A & B & C \\
\hline
1 & 2 & 3 \\
3 & 2 & 3 \\
4 & 5 & 7 \\
4 & 5 & 8 \\
\end{array}
\]

\[
\begin{array}{ccc}
A & B & \times & C & D \\
\hline
1 & 2 & & 1 & 3 \\
3 & 4 & & 2 & 4 \\
\end{array} = \\
\begin{array}{ccc}
A & B & C & D \\
\hline
1 & 2 & 1 & 3 \\
1 & 2 & 2 & 4 \\
3 & 4 & 1 & 3 \\
3 & 4 & 2 & 4 \\
\end{array}
\]

\[
\begin{array}{ccc}
A & B & C \\
\hline
1 & 2 & 3 \\
2 & 2 & 3 \\
4 & 5 & 6 \\
\end{array} \times \\
\begin{array}{ccc}
B & C & D \\
\hline
2 & 3 & 4 \\
5 & 6 & 7 \\
2 & 6 & 0 \\
\end{array} = \\
\begin{array}{ccc}
A & B & C & D \\
\hline
1 & 2 & 3 & 4 \\
2 & 2 & 3 & 4 \\
4 & 5 & 6 & 7 \\
\end{array}
\]
Query Examples

List names and cities of guests arriving on 15 May.

\[ \pi_{\text{Name, City}} \sigma_{\text{ArrivalDate} = 15 \text{ May}} (\text{g} \times \text{s}) \]

List names of each guest who has a reservation for a room that has the same name as the guest’s name.

\[ \pi_{\text{Name}} (\text{g} \times \text{s} \times \text{r}) \]

List names of guests who have a reservation for rooms with two beds.

\[ \pi_{\text{Name}} (\text{g} \times \text{s} \times \pi_{\text{RoomNr}} \sigma_{\text{NrBeds} = 2} \text{r}) \]

More Query Examples

List the names of guests from Hartford who are arriving after 10 May.

\[ \pi_{\text{Name}} (\pi_{\text{GuestNr, Name}} \sigma_{\text{City} = \text{Hartford}} \text{g} \times \text{s}) \]

List names of rooms for which no guest is arriving on 10 May.

\[ \pi_{\text{Name}} (\text{r} \times \left( \pi_{\text{RoomNr}} \text{r} - \pi_{\text{RoomNr}} \sigma_{\text{ArrivalDate} = 10 \text{ May}} \text{s} \right)) \]
SQL

Correspondence with Relational Algebra

Assume \( r(AB) \) and \( s(BC) \).

\[
\begin{align*}
\text{select } A & \quad \pi_A \sigma_B = 1 \ r \\
\text{from } r & \\
\text{where } B = 1 & \\
\text{select } B \text{ from } r & \quad \pi_B r - \pi_B S \\
\text{minus} & \\
\text{select } B \text{ from } s & \\
\text{select } A \text{ “D” from } r & \quad \rho_A \cdot D \ r \\
\text{select } A, r.B, C & \quad \pi_{A, r.B, C} \sigma_{r.B = s.B} \ (r \times s) \\
\text{from } r, s & \\
\text{where } r.B = s.B & \\
\end{align*}
\]

---

SQL: Basic Retrieval

Get full details of rooms.

\[
\begin{align*}
\text{select } \text{RoomNr, Name, NrBeds, Cost} & \quad \text{from Room} \\
\text{select } * \text{ from Room} & \\
\end{align*}
\]

Get costs.

\[
\begin{array}{c}
\text{select Cost} \\
\text{from Room} \\
\text{select distinct Cost} \\
\text{from Room} \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{COST} & \text{---------} \\
90 & \\
80 & \\
80 & \\
60 & \\
50 & \text{---------} \\
50 & \\
60 & \\
80 & \\
90 & \\
\end{array}
\]

Chapter 2 - 27

Chapter 2 - 28
SQL: Aggregate Operations

Get average cost.

```sql
select avg(Cost) from Room
```

```
<table>
<thead>
<tr>
<th>AVG(COST)</th>
</tr>
</thead>
<tbody>
<tr>
<td>72.00</td>
</tr>
</tbody>
</table>
```

```sql
select avg(Cost) "AVE$" from Room
```

```
<table>
<thead>
<tr>
<th>AVE$</th>
</tr>
</thead>
<tbody>
<tr>
<td>72.00</td>
</tr>
</tbody>
</table>
```

Also: min, max, sum, count.

---

SQL: Sorting

Get distinct costs in descending order.

```sql
select distinct Cost from Room order by Cost desc
```

```
<table>
<thead>
<tr>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>50</td>
</tr>
</tbody>
</table>
```

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

```sql
select distinct NrBeds, Cost from Room order by 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>1</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>80</td>
</tr>
</tbody>
</table>
```
SQL: Grouping

Get the average cost of rooms with the same number of beds.

```sql
select NrBeds, avg(Cost) as AVG(COST)
from Room
group by NrBeds
```

<table>
<thead>
<tr>
<th>NRBEDS</th>
<th>AVG(COST)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55.00</td>
</tr>
<tr>
<td>2</td>
<td>83.33</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>GUESTNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
</tr>
<tr>
<td>102</td>
</tr>
<tr>
<td>104</td>
</tr>
<tr>
<td>105</td>
</tr>
</tbody>
</table>

SQL: Nested Queries

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

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

<table>
<thead>
<tr>
<th>NAME</th>
<th>STREETNR</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>12 Maple</td>
<td>Boston</td>
</tr>
<tr>
<td>Carter</td>
<td>10 Main</td>
<td>Hartford</td>
</tr>
<tr>
<td>Smith</td>
<td>4 Oak</td>
<td>Providence</td>
</tr>
<tr>
<td>Green</td>
<td>10 Main</td>
<td>Boston</td>
</tr>
</tbody>
</table>
SQL: Multiple-Relation Retrieval

Get names of guests whose names match room names.

```sql
select Guest.Name
from Guest, Room
where Guest.Name = Room.Name
```

Get names of each guest who has a reservation for a room that has the same name as the guest’s name.

```sql
select g.Name
from Guest g, Room r, Reservation s
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.

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

... arriving on a date other than 15 May.

```sql
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.

```sql
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
NAME   STREETNR   CITY
------------------------------
Carter  10 Main       Hartford
Jones   6 Elm         Hartford
Smith   4 Oak         Providence
Johnson 15 Main       Boston
```

SQL: Joining a Relation with Itself

Get pairs of guest numbers of guests arriving on the same date. (The pairs should be unique.)

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

```sql
GUESTNR   GUESTNR
-------------------------------
 101       102
 101       104
 102       104
 101       105
```