What is a database

- a computer representation of the information relevant to an application.
- persistent.
- typically shared by many users and programs.

Examples:
- library database.
- credit record database.
- utilities database.
- e-commerce site database.
Database management system (DBMS)

- a software system that makes it possible to build and efficiently access large databases.
- provides a single data model.
- supports multiple languages and interfaces for data definition, manipulation, and querying.

Additional functions of a DBMS:

- consistency checking.
- concurrency control.
- resiliency.
- access control.
- meta-data.
- support for distribution, heterogeneity, multimedia, Web access, data analysis,...
Common data model

- a view of data *shared* by the programs and the users interacting with a database.
- a mathematical *abstraction*.
- supported by the DBMS.

<table>
<thead>
<tr>
<th>Data model</th>
<th>Basic notions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational</td>
<td>Relations</td>
</tr>
<tr>
<td>Object-oriented</td>
<td>Objects, classes, attributes,...</td>
</tr>
<tr>
<td>XML</td>
<td>Labelled trees (graphs)</td>
</tr>
<tr>
<td>Entity-Relationship</td>
<td>Entities, relationships,...</td>
</tr>
</tbody>
</table>
Schema vs. instance

**Schema:**
- captures and describes the structure of the data.
- time-independent.

**Instance:**
- captures the current state of the data.
- time-dependent.
- only one exists at any given time.
Relational data model

Schema concepts:

- relation schemas
- attributes
- integrity constraints

Instance concepts:

- relation instances
- tuples
- attribute values
In XML, data is self-describing, so schema is not required.

```xml
<flight>
  <number>72</number>
  <airline>Delta</airline>
</flight>
<flight>
  <airline>Delta</airline>
  <number>82</number>
  <status>canceled</status>
</flight>
```

Still, even in XML schema can be useful:

- data validation
- query optimization
Data definition (DDL):

- define database schemas.

Data manipulation (DML):

- create and update instances
- various kinds of updates:
  - incremental: “insert (1001, American) into FLIGHT”.
  - bulk: “copy all NW flights to Delta flights”.
- transactions

Query languages:

- retrieve information from database instances
Query languages

<table>
<thead>
<tr>
<th>FLIGHT</th>
<th>TERMINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER</td>
<td>AIRLINE</td>
</tr>
<tr>
<td>AIRLINE</td>
<td>TNAME</td>
</tr>
</tbody>
</table>

Simple lookup queries:

```
SELECT FLIGHT.AIRLINE
FROM FLIGHT
WHERE FLIGHT.NUMBER=72
```

Complex queries:

```
SELECT TERMINAL.TNAME
FROM FLIGHT, TERMINAL
WHERE FLIGHT.NUMBER=72
    AND FLIGHT.AIRLINE=TERMINAL.AIRLINE
```
Join

SELECT TERMINAL.TNAME
FROM FLIGHT, TERMINAL
WHERE FLIGHT.NUMBER=72
    AND FLIGHT.AIRLINE=TERMINAL.AIRLINE

<table>
<thead>
<tr>
<th>FLIGHT</th>
<th>TERMINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER</td>
<td>AIRLINE</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>Delta</td>
</tr>
<tr>
<td>82</td>
<td>Delta</td>
</tr>
<tr>
<td>1210</td>
<td>American</td>
</tr>
<tr>
<td>AIRLINE</td>
<td>TNAME</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>American</td>
<td>A</td>
</tr>
<tr>
<td>Delta</td>
<td>B</td>
</tr>
<tr>
<td>United</td>
<td>B</td>
</tr>
</tbody>
</table>
Aggregation queries

Computing aggregate values, e.g., the number of flights for each airline.

```
SELECT AIRLINE, COUNT(NUMBER)
FROM FLIGHT
GROUP BY AIRLINE
```

The result:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta</td>
<td>2</td>
</tr>
<tr>
<td>American</td>
<td>1</td>
</tr>
</tbody>
</table>
Efficient database access

Storage (disk):

- large capacity.
- persistent.
- random access.
- blocked.

Key-based access:

- primary keys (uniquely identifying):
  
  SELECT TERMINAL.TNAME
  FROM TERMINAL
  WHERE TERMINAL.AIRLINE='Delta'

- secondary keys:
  
  SELECT FLIGHT.NUMBER
  FROM FLIGHT
  WHERE FLIGHT.AIRLINE='Delta'
Levels of abstraction

Conceptual:
- the global view of the whole application.
- uses a high-level data model, e.g., relational.
- example: university database.
- created and maintained by the database administrator.

External:
- the views of users and application programs.
- uses a high-level data model, e.g., relational, XML.
- selective.
- examples: payroll, parking, registration.

Physical (internal):
- how the database is actually stored on disk.
- hidden from the users.
- uses lower-level concepts: files, indices, ...
- created and maintained by the database administrator.
Query optimization

A DBMS can evaluate a query in many different ways.

An efficient way (if one exists) is picked by the *query optimizer* module when the query is submitted, based on:

- physical schema.
- available special-purpose algorithms.
- cost analysis.
Transactions

Transaction properties:

- **Atomicity**: all-or-nothing execution
- **Consistency**: database consistency is preserved
- **Isolation**: concurrently executing transactions have no effect on one another
- **Durability**: results survive failures.