Definition: *Database design* is the activity of specifying the schema of a database in a given data model.

![Diagram of database design process](image)

**Figure 1: Process of Database Design**

**Conceptual Database Design:**

- Thinking about the data.
- Identifying semantic objects and relationships among them.
- *Goal:* Abstract but complete description of the database in a pseudo-formal notation. (paper and pencil design)
Logical Database Design:

- Transforming the initial conceptual design into a formal schema expressed in the chosen data model, i.e. deriving the conceptual schema.

Physical Database Design:

- Specifying access methods, i.e. the internal storage structures of the objects implementing the conceptual schema, i.e. deriving the internal schema.

WHY DATABASE DESIGN?

Conceptual Database Design

- Semantic clarity.

Logical Database Design

- Schema together with DML determines the application program.
- Some schemas (or a powerful DML) make queries simpler.
- With the appropriate schema one avoids redundancy, update anomalies, etc.

Physical Database Design

- Performance
- Support query optimizer.
Conceptual Database Design

The consensus choice for the data model used is the Entity-Relationship (E-R) model.

Two semantic primitives:

- **Entities**: Things that exist and are distinguishable, e.g. Paul, Sue, Sue’s car, Carnegie Hall, etc.
- **Relationships**: Two or more entities may participate in a relationship, e.g. Paul borrowed Sue’s car, Sue is a patron of Carnegie Hall.

Entities and relationships may have attributes, e.g. Paul has an age, Sue’s car has a model, Sue is a $10000 patron of Carnegie Hall.

A group of all similar (same properties) entities form an entity set. E.g. All persons, all cars, all concert halls.

Relationships can also be grouped into sets that are homogeneous in participating entity sets. Such a group is called a relationship set. E.g. All (person, car) pairs such that the person owns the car.

The overall structure of a database can be expressed graphically by an E-R diagram, which consists of the following components:

- Rectangles for entity sets.
- Diamonds for relationship sets.
- Ellipses for attributes.
- Links for associating attributes to entity and relationship sets, and entity sets to relationship sets.

**Example**: A library database has authors that have written books about various subjects (one author per book). It also has info about libraries that carry books on various subjects.

Have to identify entity and relationship sets. 1st attempt:

What is wrong? (When designing model must keep in mind what kind of questions will be asked.)

- How to represent the fact that a library carries books of a specific author.
- Where to put info about which edition of a book is carried by a library, how many copies, etc.?
Figure 2: 1st attempt E-R model

- Can we find out whether a library carries books by a certain author?

2nd attempt:

To find whether a library carries books by an author just start with the author, get his/her books and see which library they are in.

The "carry" relationship set also has attributes: how many copies, which edition.

Must also include constraint of only one author per book:  $\Rightarrow f_{\text{isunique}}$  

A relationship is an ordered set of entities. So, a relationship set is a set of these ordered sets. E.g. (Paul, Sue's car) is a member of the relationship set borrowed = { (Paul, Sue's car), (Mary, Alice's saw), (Tom, Jerry's cat) }.

A relationship may involve:

- 2 entities: binary relationship
- 3 entities: ternary relationship
- N entities: N-ary relationship
A binary relationship set between sets A and B may be:

- **1:1** Persons having social Security numbers.
- **N:1** Children having fathers. (function)
- **1:N** Fathers having children. (inverse function)
- **M:N** Students enrolling in classes.

Entities and relationships are distinguished by using various keys:

- **Definition:** A superkey is a set of one or more attributes of an entity (relationship) whose values uniquely identify an entity (relationship).

E.g. The name "Asimov", Asimov's citizenship, and Asimov's ssn is a superkey for Faulkner as an entity.
• **Definition:** A *candidate key* is a minimal superkey, i.e. no subset of its attributes is a superkey.  
  E.g. Asimov’s ssn is a candidate key.

• **Definition:** A *primary key* is one distinguished candidate key that serves as the identifier of the entity (relationship) concerned.

• **Definition:** A *foreign key* is a set of one or more attributes of an entity (relationship) that serves as a primary key of another entity or relationship.  
  E.g. For the relationship of students enrolling in classes, the student ID is a foreign key.

**Logical Data Models**

There is no standard DML for the E-R model.

Conceptual data models are mapped into logical data models.

Virtually every logical data model represents entity sets as file.

\[
\begin{align*}
\text{File} & = \text{Entity set} \\
\text{Record} & = \text{Entity} \\
\text{Field} & = \text{Attribute}
\end{align*}
\]

The differences between data models are primarily in how they represent relationships. Two basic techniques:

• Represent relationships as files as well (relational model).

• Represent relationships as links between files (network and hierarchical models).
Figure 6: File representation of Entity set

Figure 7: Relational Model Representation

Figure 8: Network Model Representation
New E/R examples

For an auto loan

Each customer may have many loans, drawn repeatedly.

Each customer has a (name, ssn, street, city, zip)

Each loan has (loan-amount, interest-rate)

Each loan has (loan-amount, balance, interest-rate)
 pouvez-vous représenter un contrat entre star - movie - studio ?
How represent a star has contract exclusively with one studio?

?  [Movie]  \rightarrow  [Star]

[Studio]

No, this means a specific star appear only once in the relationship can be done via round-about

[Movie]  \rightarrow  [In]  \rightarrow  [Star]

[Studio]  \rightarrow  [Contract]

Problem: Not all stars must have exclusive contract
Another Approach

```
*Star* <-> Contact <-> *Movie*
```

But this is really an attribute of *star-studio*
So better as

```
*Star* <-> Contact <-> *Movie*
*Star* <-> *Studio*
```
Participation constraints

I could say every study must have a president.
This is a total participation constraint
i.e. each study entity must participate in the "runs" relationship
Can a movie be owned by multiple studios?
⇒ Yes

How can you represent a contract between Star–movie–Studio?
How represent a star has contract exclusively with one studio?

No this means a specific star appear only once in the relationship.

Can be done round-about.

Problem: Not all stars must have exclusive contract.
An alternative approach:

- Star
- Studio
- Exclusive
- Movie

But this is really an attribute of star-studio

So better as:

- Star
- Studio
- Exclusive
- Contact
- Movie
- Exclusive
- Movie - Name

Studie

Studie

Studie
Participation constraints

I could say every study must have a president. This is a total participation constraint, i.e., each study entity must participate in the "runs" relationship.
Lecture 3: Relational Model

It is the most popular logical model.

Both entities and relationships are represented by relations (alias tables).

- Each entity is a single tuple (alias record, alias row) in the relation.
- Each attribute of the entity is a column (alias field) in the tuple. Recall that all entities in an entity set have the same attributes.
- Likewise, each relationship is a single tuple in the relation.
- The fields in a relationship relation are the primary keys of the participating entities plus any additional attributes the relationship might have.

![Diagram of Library E-R Model](image)

Figure 1: Library E-R model

Optimization: Functional relationships do not need a separate relation to be represented. For a functional relationship from set A to set B, incorporate the primary key of B into A.
E.g. If every book has only one author, the wrote is a functional relationship. In the relational schema, get rid of the relation wrote and add ssn as an attribute of book. This save one relation.

General comments:

- A relation is a set of tuples.
- Order of tuples in a relation is not important.
- Order of attributes is important. To avoid this we give names to attributes.
- Every tuple is stored only once in a relation.
- A value may appear multiple times in a column. If a single attribute is the primary key for a relation, can a value appear multiple times in the column corresponding to that attribute?
Figure 3: Set Model Approach

We could represent all relationships using set-valued attributes instead of atomic values:

Problems:

- non-standard width for fields
- long search time
- complicated DML (2nd order logic)

Relational model only allows fixed width fields.

Extensions to the Relational Model

- Referential Integrity: E.g. books must have an author.

  Ex 2: Insert employee sue assuming works in toy dept. No toy dept in dept relation!

<table>
<thead>
<tr>
<th>emp</th>
<th>name</th>
<th>id</th>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>sue</td>
<td>22</td>
<td>toy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dept</th>
<th>name</th>
<th>floor</th>
</tr>
</thead>
<tbody>
<tr>
<td>shoe</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>candy</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>sports</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: Referential Integrity Example

- Generalization Hierarchy: E.g. my honda is a car, cars are vehicles, vehicles are physical objects.
- Order Relations: E.g. if some text is stored in a relation, word by word, there is an implicit order among the tuples.
Lecture 4: Relational Algebra

Query Languages are either:

- Procedural
- Nonprocedural

Relational Algebra is a formal procedural query language.

The language is composed of five fundamental operators:

\[ \{ \sigma, \pi, \times, \cup, \neg \} \]

Assume the following schemas:

Student(name,sid,ssn,status)
Faculty(name,fid,ssn,salary)
Courses(name,credits,dept, cnum,fid)
Grades(sid,cnum,semester,grade)
Enrolled(sid,cnum)

<table>
<thead>
<tr>
<th>student</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>sid</td>
<td>ssn</td>
<td>status</td>
</tr>
<tr>
<td>Bob</td>
<td>99</td>
<td>121</td>
<td>4</td>
</tr>
<tr>
<td>Sue</td>
<td>19</td>
<td>934</td>
<td>3</td>
</tr>
<tr>
<td>John</td>
<td>12</td>
<td>813</td>
<td>3</td>
</tr>
<tr>
<td>Mary</td>
<td>73</td>
<td>395</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>faculty</th>
<th></th>
<th></th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>fid</td>
<td>ssn</td>
<td></td>
</tr>
<tr>
<td>Lent</td>
<td>9</td>
<td>912</td>
<td>1</td>
</tr>
<tr>
<td>Nicol</td>
<td>7</td>
<td>812</td>
<td>81</td>
</tr>
<tr>
<td>Park</td>
<td>3</td>
<td>452</td>
<td>105</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>courses</th>
<th>credits</th>
<th>dept</th>
<th>cnum</th>
<th>fid</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>3</td>
<td>cs</td>
<td>521</td>
<td>9</td>
</tr>
<tr>
<td>simulation</td>
<td>5</td>
<td>cs</td>
<td>426</td>
<td>7</td>
</tr>
<tr>
<td>architecture</td>
<td>3</td>
<td>cs</td>
<td>314</td>
<td>3</td>
</tr>
</tbody>
</table>
### Lecture 4

#### Grades

<table>
<thead>
<tr>
<th>sid</th>
<th>cnum</th>
<th>semester</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>314</td>
<td>92.1</td>
<td>A</td>
</tr>
<tr>
<td>99</td>
<td>426</td>
<td>92.2</td>
<td>B</td>
</tr>
<tr>
<td>73</td>
<td>521</td>
<td>92.3</td>
<td>A</td>
</tr>
<tr>
<td>73</td>
<td>426</td>
<td>92.3</td>
<td>A</td>
</tr>
<tr>
<td>73</td>
<td>314</td>
<td>92.3</td>
<td>A</td>
</tr>
<tr>
<td>12</td>
<td>521</td>
<td>93.1</td>
<td>C</td>
</tr>
</tbody>
</table>

#### Enrolled

<table>
<thead>
<tr>
<th>sid</th>
<th>cnum</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>321</td>
</tr>
<tr>
<td>99</td>
<td>111</td>
</tr>
<tr>
<td>19</td>
<td>321</td>
</tr>
<tr>
<td>82</td>
<td>101</td>
</tr>
<tr>
<td>73</td>
<td>321</td>
</tr>
</tbody>
</table>

### Selection (\(\sigma\))

Selects tuples that satisfy given predicate:

\[
\sigma_{\text{name}="lect"}(\text{faculty})
\]

\[
\sigma_{\text{name}="Hob"}(\text{students})
\]

\[
\sigma_{\text{status}>3}(\text{students})
\]

\[
\sigma_{\text{credits}>3 \land \text{dept}="cs"}(\text{courses})
\]

\[
\sigma_{\text{cnum}>500}(\text{enrolled})
\]

\[
\sigma_{\text{sid}=\text{sen}}(\text{student})
\]

Say only want name of seniors.

### Projection (\(\pi\))

Copies relation with only columns given as arguments:

\[
\pi_{\text{name}, \text{sid}}(\text{student})
\]
Lecture 4

<table>
<thead>
<tr>
<th>Bob</th>
<th>99</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sue</td>
<td>19</td>
</tr>
<tr>
<td>John</td>
<td>12</td>
</tr>
<tr>
<td>Mary</td>
<td>73</td>
</tr>
</tbody>
</table>

\[ \pi_{\text{salary}}(\text{faculty}) \]

1
81
105

Names of seniors:

\[ \pi_{\text{name}}(\sigma_{\text{status}>3}(\text{students})) \]

Say now want names of students enrolled in 321. Must have operator that takes two relations, student and enrolled, as arguments.

Cartesian Product \((\times)\).

Takes the cross product off two relations, i.e. each tuple in the second relation is concatenated to each tuple in the first relation.

\[ r = \text{student} \times \text{enrolled} \]

Schema of resultant relation is all the attributes of both relations:

schema of \( r = (\text{student.name}, \text{student.sid}, \text{student.ssn}, \text{student.status}, \text{enrolled.sid}, \text{enrolled.cnum}) \);

How get names of all students enrolled in 321?

\[ \pi_{\text{student.name}}(\sigma_{\text{enrolled.cnum}=321}(\text{students} \times \text{enrolled})) \]

Is this correct? NO

\( \text{student} \times \text{enrolled} \):
<table>
<thead>
<tr>
<th>student.name</th>
<th>student.sid</th>
<th>student.ssn</th>
<th>student.status</th>
<th>enrolled.sid</th>
<th>enrolled.cnum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>99</td>
<td>121</td>
<td>4</td>
<td>99</td>
<td>321</td>
</tr>
<tr>
<td>Bob</td>
<td>99</td>
<td>121</td>
<td>4</td>
<td>99</td>
<td>111</td>
</tr>
<tr>
<td>Bob</td>
<td>99</td>
<td>121</td>
<td>4</td>
<td>19</td>
<td>321</td>
</tr>
<tr>
<td>Bob</td>
<td>99</td>
<td>121</td>
<td>4</td>
<td>82</td>
<td>101</td>
</tr>
<tr>
<td>Bob</td>
<td>99</td>
<td>121</td>
<td>4</td>
<td>73</td>
<td>321</td>
</tr>
<tr>
<td>Sue</td>
<td>19</td>
<td>934</td>
<td>3</td>
<td>99</td>
<td>321</td>
</tr>
<tr>
<td>Sue</td>
<td>19</td>
<td>934</td>
<td>3</td>
<td>99</td>
<td>111</td>
</tr>
<tr>
<td>Sue</td>
<td>19</td>
<td>934</td>
<td>3</td>
<td>19</td>
<td>321</td>
</tr>
<tr>
<td>Sue</td>
<td>19</td>
<td>934</td>
<td>3</td>
<td>82</td>
<td>101</td>
</tr>
<tr>
<td>Sue</td>
<td>19</td>
<td>934</td>
<td>3</td>
<td>73</td>
<td>321</td>
</tr>
<tr>
<td>John</td>
<td>12</td>
<td>813</td>
<td>3</td>
<td>99</td>
<td>321</td>
</tr>
<tr>
<td>John</td>
<td>12</td>
<td>813</td>
<td>3</td>
<td>99</td>
<td>111</td>
</tr>
<tr>
<td>John</td>
<td>12</td>
<td>813</td>
<td>3</td>
<td>19</td>
<td>321</td>
</tr>
<tr>
<td>John</td>
<td>12</td>
<td>813</td>
<td>3</td>
<td>82</td>
<td>101</td>
</tr>
<tr>
<td>John</td>
<td>12</td>
<td>813</td>
<td>3</td>
<td>19</td>
<td>321</td>
</tr>
<tr>
<td>Mary</td>
<td>73</td>
<td>395</td>
<td>4</td>
<td>99</td>
<td>321</td>
</tr>
<tr>
<td>Mary</td>
<td>73</td>
<td>395</td>
<td>4</td>
<td>99</td>
<td>111</td>
</tr>
<tr>
<td>Mary</td>
<td>73</td>
<td>395</td>
<td>4</td>
<td>19</td>
<td>321</td>
</tr>
<tr>
<td>Mary</td>
<td>73</td>
<td>395</td>
<td>4</td>
<td>82</td>
<td>101</td>
</tr>
<tr>
<td>Mary</td>
<td>73</td>
<td>395</td>
<td>4</td>
<td>73</td>
<td>321</td>
</tr>
</tbody>
</table>

Result of query would be Bob,Bob,Bob,Sue,Sue,Sue,John,John,John,Mary,Mary,Mary

Since relations are sets, and hence duplicates are removed, result is Bob,Sue,John, Mary. Note, John is not enrolled in 321.

How correct the query?

\[
\pi_{\text{student.name}}(\sigma_{\text{student.sid} = \text{enrolled.sid}}(\sigma_{\text{enrolled.cnum} = 321}(\text{students} \times \text{enrolled})))
\]

How find name of all faculty who are teaching graduate courses?

\[
\pi_{\text{faculty.name}}(\sigma_{\text{faculty.fid} = \text{courses.fid}}(\sigma_{\text{courses.cnum} > 500}(\text{faculty} \times \text{courses})))
\]
<table>
<thead>
<tr>
<th>fac.name</th>
<th>fac.fid</th>
<th>fac.ssn</th>
<th>fac.salary</th>
<th>c.name</th>
<th>c.credits</th>
<th>c.dept</th>
<th>c.cnum</th>
<th>c.fid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leut</td>
<td>9</td>
<td>912</td>
<td>1</td>
<td>database</td>
<td>3</td>
<td>cs</td>
<td>521</td>
<td>9</td>
</tr>
<tr>
<td>Leut</td>
<td>9</td>
<td>912</td>
<td>1</td>
<td>simulation</td>
<td>5</td>
<td>cs</td>
<td>426</td>
<td>7</td>
</tr>
<tr>
<td>Leut</td>
<td>9</td>
<td>912</td>
<td>1</td>
<td>architecture</td>
<td>3</td>
<td>cs</td>
<td>314</td>
<td>3</td>
</tr>
<tr>
<td>Nicol</td>
<td>7</td>
<td>812</td>
<td>81</td>
<td>database</td>
<td>3</td>
<td>cs</td>
<td>521</td>
<td>9</td>
</tr>
<tr>
<td>Nicol</td>
<td>7</td>
<td>812</td>
<td>81</td>
<td>simulation</td>
<td>5</td>
<td>cs</td>
<td>426</td>
<td>7</td>
</tr>
<tr>
<td>Nicol</td>
<td>7</td>
<td>812</td>
<td>81</td>
<td>architecture</td>
<td>3</td>
<td>cs</td>
<td>314</td>
<td>3</td>
</tr>
<tr>
<td>Park</td>
<td>3</td>
<td>452</td>
<td>105</td>
<td>database</td>
<td>3</td>
<td>cs</td>
<td>521</td>
<td>9</td>
</tr>
<tr>
<td>Park</td>
<td>3</td>
<td>452</td>
<td>105</td>
<td>simulation</td>
<td>5</td>
<td>cs</td>
<td>426</td>
<td>7</td>
</tr>
<tr>
<td>Park</td>
<td>3</td>
<td>452</td>
<td>105</td>
<td>architecture</td>
<td>3</td>
<td>cs</td>
<td>314</td>
<td>3</td>
</tr>
</tbody>
</table>

How get names of all students who have failed at least one course?

\[ \pi_{\text{student.name}}(\sigma_{\text{student.sid} = \text{grades.sid}}(\sigma_{\text{grades.grade} = F}(\text{student } \times \text{grades}))) \]

Suppose want to know all students who are taking 321 or 444?

**Union (\( \cup \))**

\( r_1 \cup r_2 \) is the union of relations \( r_1 \) and \( r_2 \).

\[ r_1 = \pi_{\text{student.name}}(\sigma_{\text{student.sid} = \text{enrolled.sid}}(\sigma_{\text{enrolled.cnum} = 321}(\text{student } \times \text{enrolled}))) \]

\[ r_2 = \pi_{\text{student.name}}(\sigma_{\text{student.sid} = \text{enrolled.sid}}(\sigma_{\text{enrolled.cnum} = 444}(\text{student } \times \text{enrolled}))) \]

Answer: \( r_1 \cup r_2 \)

**Set Difference (\( - \))**

\( r_1 - r_2 \) is those members of \( r_1 \) that are not in \( r_2 \).

Say want students who are taking 321 but not 444:

\[ r_1 - r_2. \]