Integrated Database Design

Mark Graves

Agenda: Integrated Database Design

- Definitions & Overview
- Graph Conceptual Design
- XML Document Design
- Relational Database Design
- Summary

Relational Databases

- Relational Database -- a collection of relations
- Example

  Person(name, phone, email)

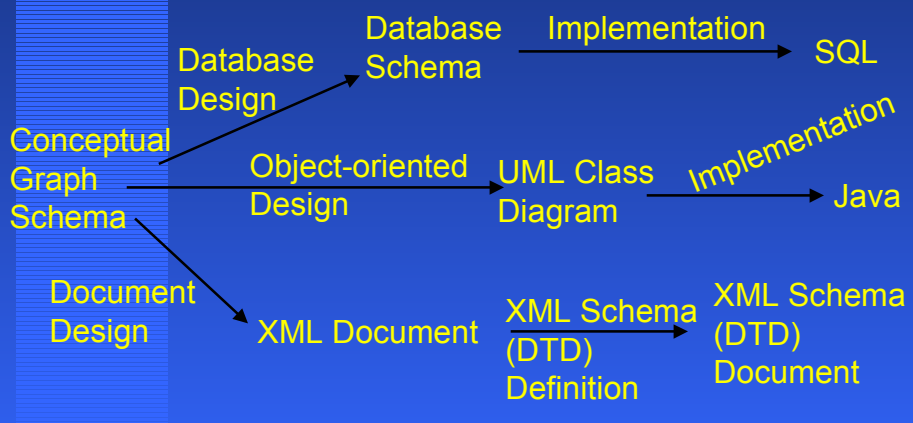
  Gene(name, organism, chr_loc)
- Table -- a data structure for implementing a relation
- Implement using SQL

```
create table gene (
    id          number,
    name        string,
    organism    string,
    chr_loc     string);
```

Database Definitions

- Database -- collection of persistent data
- Database Management System -- software to manage a database (add, delete, modify, query)
- Database System -- database, DBMS, application software, users, hardware, …
- Web-enabled Database -- database accessible via WWW (internet or intranet)
- Web-enabled Database System -- database, users, DBMS, web servers, application software, and other software to deliver persistent data via WWW

Integrated Design Process



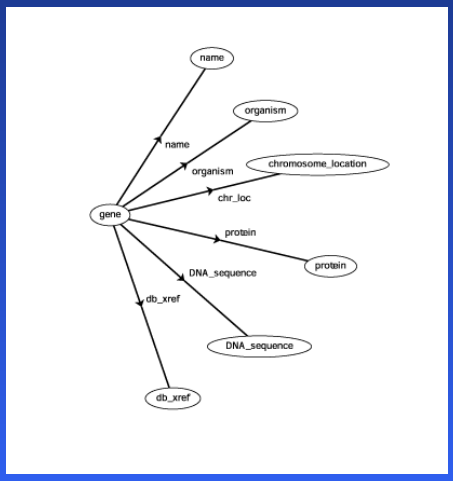
Benefits

- Unified Design
- Smooth data exchange between:
  - Relational Databases
  - Object-oriented applications
  - XML documents

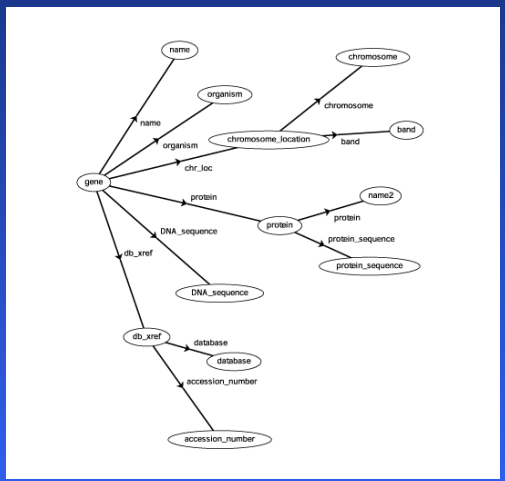
Graph Conceptual Model

- Graph is a collection of nodes and edges
- Node represents
  - concept
  - object
  - complex relationship
- Edge represents
  - characteristic
  - binary relationship

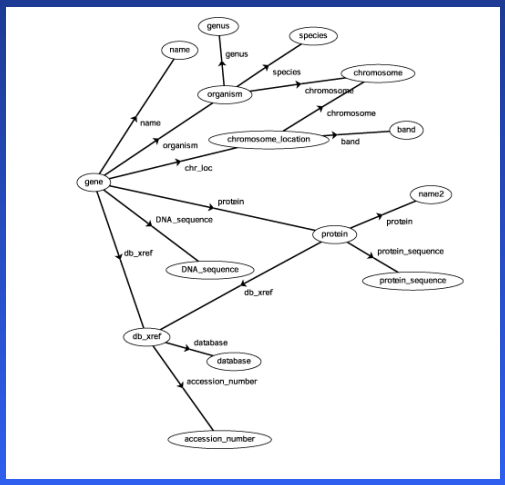
Example Conceptual Schema



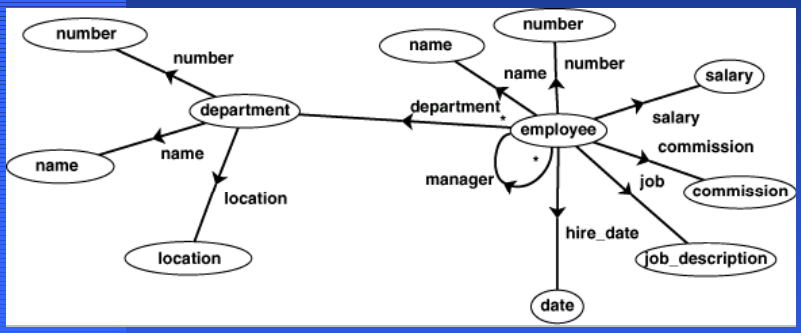
Example Conceptual Schema



Example Conceptual Schema



Example Conceptual Schema



Graph Conceptual Design

1. List domain concepts and relationships.
2. Connect domain concepts by simple sentences (that describe relationships).
3. Select major concepts in the domain from the list.
4. Draw the major concepts as nodes.
5. Draw the simple sentences as edges.
6. Add cardinality constraints to the edges.
7. Revise the schema.

Basic Gene Concepts

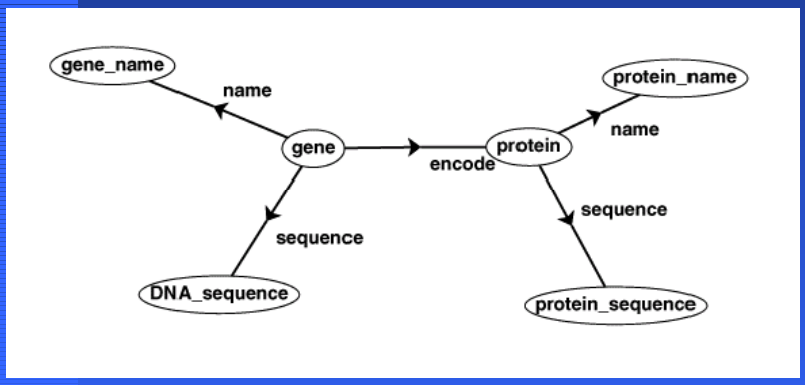
Concepts

- Gene
- Protein
- Gene name
- Gene sequence
- Protein name
- Protein sequence

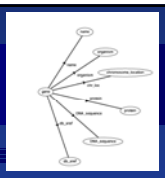
Simple Sentences

- Gene has a gene name.
- Gene has a gene sequence.
- Protein has a protein name.
- Protein has a protein sequence.
- Gene encodes for a protein.

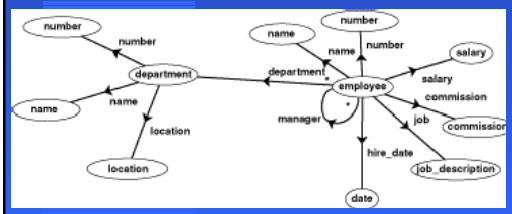
Gene Conceptual Schema

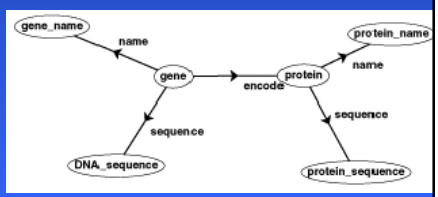


Exercise



1. List concepts and relationships.
2. Connect concepts (or relationships) by simple sentences.
3. Select major concepts from the list.
4. Draw major concepts as nodes.
5. Draw simple sentences as edges.





Microarray Definition

- Microarray hybridization experiments consist of labeled DNA probes being hybridized against DNA pieces that have been immobilized in a grid layout on a glass slide.
- Spots on the slide 'light up' when a probe DNA sticks to one of the immobilized DNA spots.
- Experimental results describe the intensity of each spot on the grid.

Microarray Concepts & Relationships

Concepts

- **Experiment, Gene, Array, Spot**

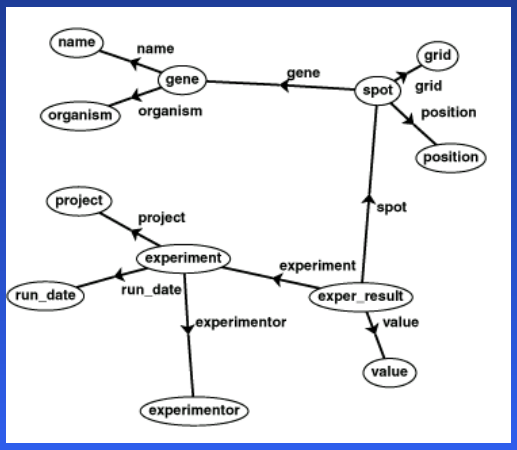
Complex Relationship

- **Experiment has a result for a spot. (ExperimentResult)**

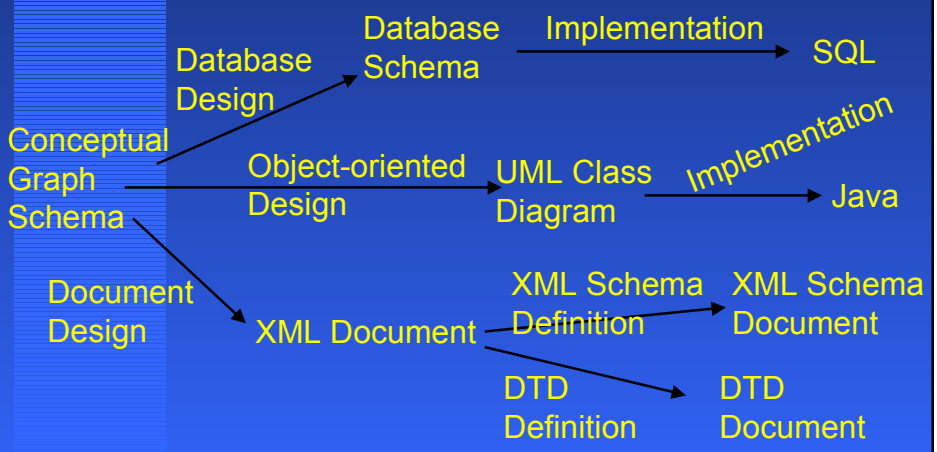
Simple Sentences

- **Experiment has conditions.**
- **Gene has a name.**
- **Array has a collection of spots.**
- **Spot has a gene.**
- **Experiment has an experiment result**
- **Experiment Result has a value**
- **Experiment Result has a spot**

Conceptual Schema



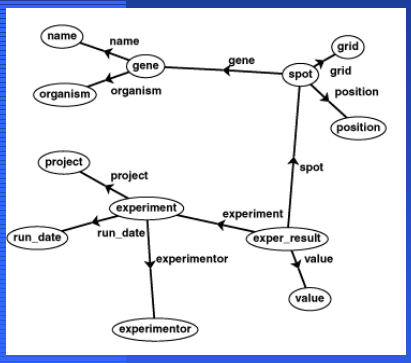
Integrated Design Process



XML Document Design Process

- Define Empty XML Document (with Elements & Attributes)
  - Choose graph node for root element(s).
  - Each graph node with protruding edges becomes an element type with the same name.
  - When a edge refers to a concept with no protruding edges it may be either an attribute or a subelement as preferred, with the following recommendations:
    - If a characteristic depends on the concept for its existence in the domain and has cardinality 1:1 or Many:1, it becomes an attribute.
    - If more detail may be added later, then it should be a subelement.
- Translate Empty XML Document to DTD or XML Schema
- Add restrictions for cardinality and type

Define Empty XML Document



Empty XML

```
<experiment>
  <project/>
  <run_date/>
  <experimentor/>
  <exper_result value="">
    <spot grid="" position="">
      <gene>
        <name/>
        <organism/>
      </gene>
    </spot>
  <exper_result>
</experiment>
```

XML Schema

Empty XML

```
<experiment>
  <project/>
  <run_date/>
  <experimentor/>
  <exper_result value="">
    <spot grid="" position="">
      <gene>
        <name/>
        <organism/>
      </gene>
    </spot>
  <exper_result>
</experiment>
```

XML Schema

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2000/10/XMLSchema">
<element name="experiment"  type="experimentType"/>
<complexType name="experimentType">
   <element name="project"/>
   <element name="run_date" type="date"/>
   <element name="experimentor"/>
   <element name="exper_result" type="exper_resultType"/>
</complexType>
<complexType name="exper_result">
   <attribute name="value"/>
   <element name="spot" type="spotType"/>
</complexType>
<complexType name="spot">
   <attribute name="grid"/>
   <attribute name="position"/>
   <element name="gene" type="geneType"/>
<complexType name="gene">
   <element name="name">
   <element name="organism"/>
</complexType>
</schema>
```

Document Type Definition

Empty XML

```
<experiment>
  <project/>
  <run_date/>
  <experimentor/>
  <exper_result value="">
    <spot grid="" position="">
      <gene>
        <name/>
        <organism/>
      </gene>
    </spot>
  <exper_result>
</experiment>
```

DTD

```
<!ELEMENT experiment (project, run_date,
                      experimentor, exper_result*)>
<!ELEMENT project CDATA>
<!ELEMENT run_date CDATA>
<!ELEMENT experimentor CDATA>
<!ELEMENT exper_result (spot)>
<!ATTLIST exper_result
  value      CDATA #REQUIRED>
<!ELEMENT spot (gene)>
<!ATTLIST spot
  grid       CDATA #REQUIRED
  position  CDATA #REQUIRED>
<!ELEMENT gene (name, organism)>
<!ELEMENT name CDATA>
<!ELEMENT organism CDATA>
```

Exercise

- Choose concept for root element(s)
- Each concept with edges becomes an element
- When a edge refers to a concept with no edges:
  - If a characteristic depends on the concept for its existence in the domain and has cardinality 1:1 or Many:1, it becomes an attribute.
  - If more detail may be added later, then it should be a subelement.

```
<genes>
  <gene id="">
    <name/>
    <organism/>
    <chr_loc chr=""/>
    <protein id=""/>
    <DNA_sequence/>
    <db_xref gi=""/>
  </gene>
</genes>

<experiment>
  <project/>
  <run_date/>
  <experimentor/>
  <exper_result value="">
    <spot grid="" position="">
      <gene>
        <name/>
        <organism/>
      </gene>
    </spot>
  <exper_result>
</experiment>
```

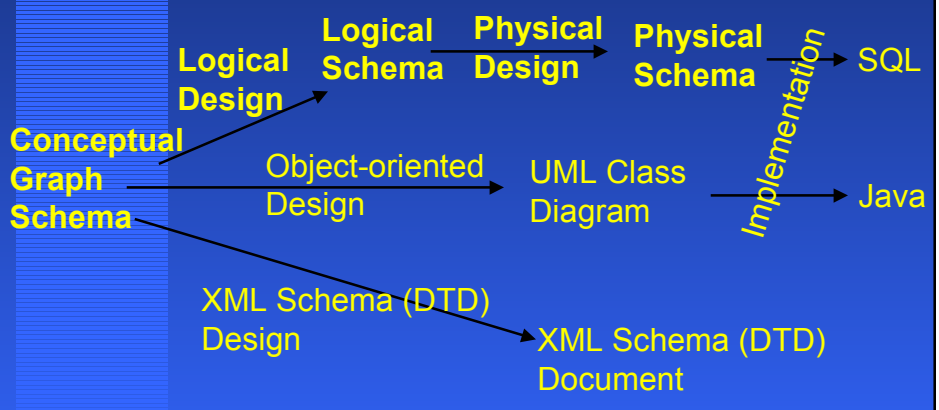
Database Design

- **Conceptual Design -- Unconstrained description of the data to gain overview of data**
- **Logical Design -- Precise, mathematical description of data to understand types and relationships of data**
- **Physical Design -- Specification in Data Definition Language of selected DBMS to specify all data requirements for implementation**

Database Design Schemas

- **Conceptual Schema -- Language of the domain**
- **Logical Schema -- Language of relational databases**
- **Physical Schema -- Language of a relational DBMS**

Integrated Design Process



Relational Database Design

- **Graph Conceptual Design**
- **Logical Design (embedded relations)**
- **Logical Design (1st normal form)**
- **Physical Design**
- **Implementation**

Logical Design

- Each concept that aggregates other concepts becomes a relation.
- Each concept without characteristics becomes an atomic value (implied or explicit).
- Other concepts become non-relational domains.

Logical Schema

- Gene(name STRING, organism ORGANISM)
- Spot(gene GENE, grid GRID, position GRID_POSITION)
- Experiment(project STRING, run_date DATE, experimenter PERSON)
- Experiment_result(experiment EXPERIMENT, variant STRING, spot SPOT, value NUMBER)

Normal Forms

- **Remove embedded relations (first normal form)**
- **Create unique identifiers (primary keys) for each relation if needed**
- **Create foreign key constraints to associate use of identifiers (foreign keys) with their relation (primary keys)**
- **Some cardinalities (many-to-many) may require helper relations.**

Normalized Logical Schema

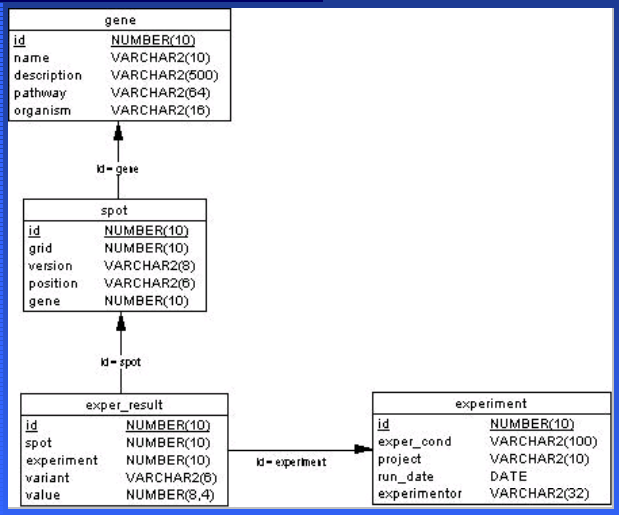
- Gene(id GENE_ID, name STRING, organism ORGANISM)
- Spot(id SPOT_ID, gene GENE_ID, grid GRID, position GRID_POSITION)
- Experiment(id EXPERIMENT_ID, project STRING, run_date DATE, experimenter PERSON)
- Experiment_result(experiment EXPERIMENT_ID, variant STRING, spot SPOT_ID, value NUMBER)

⇨ Every additional use of GENE_ID, SPOT_ID, EXPERIMENT_ID is a foreign key reference

Physical Design

- **Define the physical data type for each domain that is not associated with a relation**
- **Define additional database constraints for uniqueness and cardinality**

Physical Schema

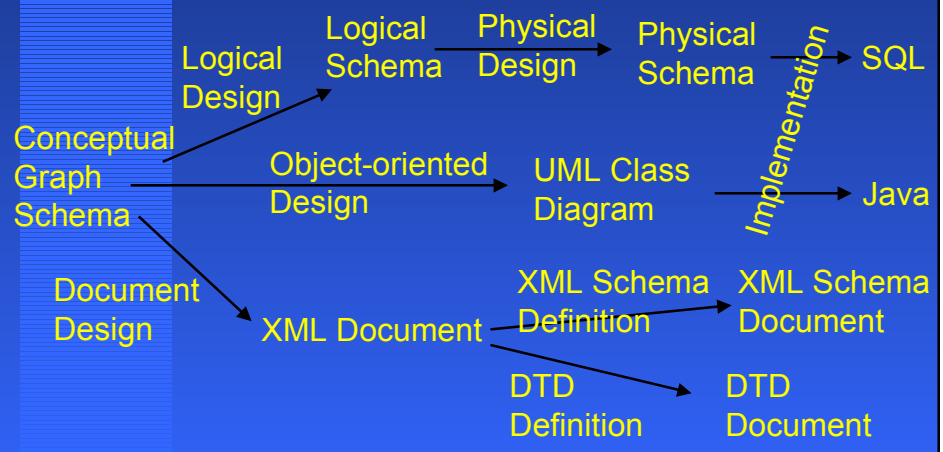


Implementation

```
create table gene (
  id              number(10) not null,
  name            varchar2(10),
  description     varchar2(500),
  pathway         varchar2(64),
  organism        varchar2(16));
create table spot (
  id              number(10) not null,
  grid            number(10) not null,
  version         varchar2(8),
  position        varchar2(6) not null,
  gene            number(10) not null);
create table experiment (
  id              number(10) not null,
  exper_cond      varchar2(100),
  project         varchar2(10),
  run_date        date,
  experimentor    varchar2(32));
```

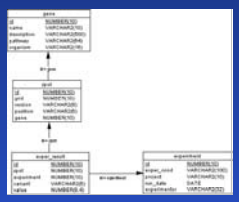
```
create table exper_result (
  id              number(10) not null,
  spot            number(10),
  experiment      number(10) not null,
  variant         varchar2(6),
  value           number(8,4));

-- Constraints

alter table gene add primary key (id);
alter table spot add primary key (id);
alter table experiment add primary key (id);
alter table exper_result add primary key (id);
alter table spot add constraint fk1_gene
  foreign key (gene) references gene (id);
alter table exper_result add constraint fk1_exper_result
  foreign key (spot) references spot (id);
alter table exper_result add constraint fk2_exper_result
  foreign key (experiment) references experiment (id);
```

Integrated Design Process



Micro-array Design Deliverables

Physical Schema



Logical Schema

Gene(name STRING, organism ORGANISM)
Spot(gene GENE, grid GRID, position GRID_POSITION)
Experiment(project STRING, run_date DATE, experimenter PERSON)
Experiment_result(experiment EXPERIMENT, variant STRING, spot SPOT, value NUMBER)

Conceptual Graph Schema

Empty XML

```
<experiment>
  <project/>
  <run_date/>
  <experimentor/>
  <exper_result value="">
    <spot grid="" position="">
      <gene>
        <name/>
        <organism/>
      </gene>
    </spot>
  <exper_result>
</experiment>
```

XML Schema

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2000/10/XMLSchema">
<element name="experiment"  type="experimentType"/>
<complexType name="experimentType">
   <element name="project"/>
   <element name="run_date" type="date"/>
   <element name="experimentor"/>
   <element name="exper_result" type="exper_resultType"/>
</complexType>
<complexType name="exper_result">
   <attribute name="value"/>
   <element name="spot" type="spotType"/>
</complexType>
<complexType name="spot">
   <attribute name="grid"/>
   <attribute name="position"/>
   <element name="gene" type="geneType"/>
<complexType name="gene">
   <element name="name">
   <element name="organism"/>
</complexType>
</schema>
```