Course Overview

1. Data Integration and …
   - structured (relational) databases
   - knowledge-based extensions, ontologies
   - semi-structured (XML) databases

2. Scientific Workflows
   - Dataflow process networks
   - Web service workflows
   - The Kepler system

3. Student projects on (1) and (2)

Perfect Recall: Database Systems (165A)

- A Database System (DBS) consists of a Database (DB) and a Database Management System (DBMS)
- A Database is a (typically very large) integrated collection of interrelated data which are stored in files.
- Data can come from commercial or scientific applications and (usually) represent some abstraction/piece of the modeled real world.
- E.g., a scientific database might contain information about known biological, chemical, astronomical entities, lab experiments, etc
- A Database Management System is a collection of software packages designed to store, access, and manage databases. It provides users and applications with an environment that is convenient and efficient to use.
Relational Database Model

- Think of a relational DB as a number of tables, each have a particular schema:
  - Course(Instructor, Name, Quarter, Department)
- The table/relation name “Course”, identifies which table we are talking about.
- The attribute/column name (e.g., “Instructor”) corresponds to the “column header”
- Elements aka instances or tuples of a table/relation can be written, e.g., as follows:
  
```plaintext
Course("Gertz", "ECS165A", "W-2005", "CS").
Course("Ludaescher", "ECS289F", "W-2005", "CS").
```

Example

<table>
<thead>
<tr>
<th>Course</th>
<th>Instructor</th>
<th>Name</th>
<th>Quarter</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gertz</td>
<td>ECS165A</td>
<td>W-2005</td>
<td>CS</td>
</tr>
<tr>
<td></td>
<td>Ludaescher</td>
<td>ECS289F</td>
<td>W-2005</td>
<td>CS</td>
</tr>
</tbody>
</table>

- The same in **Datalog notation** – as a set of facts:
  
```plaintext
course('Ludaescher', 'ECS289F', 'W-2005', 'CS').
course(\ldots, \ldots, \ldots).
```

Hmm.. looks like a Spreadsheet ...

- … but there are differences.
- What are they?
Data Integration (Mediator System)

Integrated Global (XML) View

Integrated View

Definition

G(...) ← S1(...)…Sk(...) ...

USER/Client

1. Query Q ( G(S1, …, Sk) )

MEDIATOR

2. Query processing

3. Q1

Q2

Q3

4. {answers(Q1)}

{answers(Q2)}

{answers(Q3)}

5. Post processing

6. {answers(Q)}

web services as wrapper APIs

Query Languages

- Databases can be **queried**!
- We state a question, usually in terms of the given database schema, about the stored data.
- **Query languages** such as Datalog and SQL (Structured Query Language) are **declarative** (just say what you’re interested in) – you do not need to give the details *how* to retrieve the data, but can focus on the *what* (to retrieve).

Question

- What’s the difference between **keyword-based search** and **querying a database**?

- But watch out
  – … some recent work in the database community on “keyword search in databases”…
What is a Query?

- A **query expression** e.g. in SQL or in Datalog denotes a query (but we still don’t know what a query is...)
- A **query** is a (generic*) **mapping** \( f \) from instances of an input schema (EDB) to instances of an output schema (IDB):
  \[
  f : \text{inst(EDB)} \rightarrow \text{inst(IDB)}
  \]
- Note: Different query expressions can denote the same query (mapping). Example...?
**What is a Query?**

- A query is a **generic mapping** $f$ from instances of an input schema ($EDB$) to instances of an output schema ($IDB$):
  
  ![](image)

- **generic**: invariant under renamings $r$, i.e., $f(r(I)) = r(f(I))$ for all database instances $I$ of the schema $EDB$

- Examples: Consider $EBD = \{p(X), emp(N,S)\}$. Which of the following are generic?
  - $f$-even: "T" if $\{|x| p(x) \text{ is in } DB I \text{ }| \text{ is even}
  - $f$-jeff: $\{(N,S) | emp(N,S) \text{ in } DB I, N = \text{"Jeff"} \}$

**Problem**

- How can one **evaluate** DATALOG queries? That is, given a database instance (= a set of facts), how can one obtain the answer to a given query (=rule or set of rules)?

**DATALOG: Fixpoint Semantics (Bottom-Up)**

Relations can be defined (directly or indirectly) in terms of themselves (= recursive definitions/rules):

- $e(a,b), e(b,c), \ldots$ % FACTS (EDB-relations: $e$)
- $t(x,y) = e(x,x)$ % RULES (EDB-relations: $t$)
- $t(x,y) = e(x,y), t(x,y)$. % RULES (EDB-relations: $t$)

- **Bottom-Up Evaluation (Fixpoint Semantics)**: Apply rules iteratively (in so-called $T$- rounds) until a fixpoint is reached ($P = \text{Facts} \cup \text{Rules}$):

  ![](image)

  The sequence $I_0 \subseteq I_1 \subseteq I_2 \ldots$ converges to the least fixpoint $I_f(T_f) \text{ of } T_f$

  $T_f$ (Immediate Consequences) operator

  $T_f(P) = \{ \text{Head} | \text{Head} \leftrightarrow \text{Body} \in P, I \models \text{Body} \}$
Example: Transitive Closure

\[
\begin{align*}
\text{\texttt{e(a,b). e(b,c). e(c,d).}} & \quad \% \text{FACTS (EDB-relation: e/2)} \\
\text{tc(X,Y) :- e(X,Z). tc(Z,Y).} & \quad \% \text{RULES (EDB-relation: tc/2)} \\
\text{tc(X,Y) :- tc(X,Z), tc(Z,Y).} & \\
\end{align*}
\]

\(\text{n} \rightarrow 0 \text{ if } \text{length of path is 0} \)

\(\text{Exercise: How about the following rule?} \quad \text{tc(X,Y)} \quad \leftarrow \text{tc(X,Z), tc(Z,Y).} \)

DATALOG: Minimal Model Semantics

- Rules can be seen as first-order formulas:
\(p(x) \leftarrow q(x,y) \land \forall x q(x,y) \lor \exists y q(x,y))\)
- A model of a program \(P := \text{Fact} \cup \text{Rules}\) is an interpretation of the relations which satisfies all rules.
(Here, we're dealing with Herbrand interpretations and models, i.e., which interpret constants "syntactically". Then, an interpretation can be written as a set of true tuples, all other tuples are regarded as false.)
- From the possible different models find the "smallest", "least" model, i.e., which makes true only what is "strictly necessary", given the rules of \(P \rightarrow \) minimal model.
- A model \(M\) of \(P\) is minimal, if there is no other model \(M' \subseteq M\).

Query Languages for Relational Databases

- SQL
- Relational Algebra (RA)
- Relational Calculus (RC)
- Datalog

EXAMPLE Given relations: employee(Employee, Salary, DeptNo) and dept(DeptNo, Mgr)

find all (employee, manager) pairs.

- SQL: SELECT Emp, Mgr
  FROM employee, dept
  WHERE employee.DeptNo = dept.DeptNo
- RA: \(\pi_{\text{Emp, Mgr}}(\text{employee}) \land \text{dept} = \text{DeptNo})
- RC: \(F(\text{Emp, Mgr}) = \text{Salary} = \text{Employee}(\text{Emp, Salary, DeptNo}) \land \text{DeptNo} = \text{Mgr})
- Datalog: boss(Emp, Mgr) \leftarrow \text{employee}(Emp, Salary, DeptNo), dept(DeptNo, Mgr)