Information and Database Management Principles

- ***Read for course information at the beginning***
- ***Sakai for***
  - *access to lecture notes (when available)*
  - *announcements*
  - *homework postings/submissions, … {please verify that your **submission** are actually there in sakai}*
  - *project postings/submissions*
  - *resources*

Intro to Databases (J.Widom - Stanford)

Database Management System (DBMS) provides….

… efficient, reliable, convenient, and safe multi-user storage of and access to massive amounts of persistent data.

Information Management in the World

- **Motivating example:**
  "Find talks given by Rutgers DCS faculty in 2015"
- **Where would you have to look?**
  - "brain"
  - paper records (calendars)

  COMPUTER:
  - plain text files
  - Google/... calendar
  - UNIX *calendar* file -- gives warnings for lines with "jul 24"
  - spreadsheet
  - database management system
  - web page

Topics

What kinds of information are there?

- **structured data:**
  - relational DBMS; SQL queries; triggers; integrity constraints
  - web interfaces accessing relational dbms from Java;
- **semi-structured data:**
  - XML: storage and querying (Xpath,Xquery)
  - the Web: Google and other page ranking algorithms
- **unstructured data: text information retrieval with vector space model**
- **knowledge: conceptual models in ER and UML; deduction in logic; (ontologies/graph dbs)**

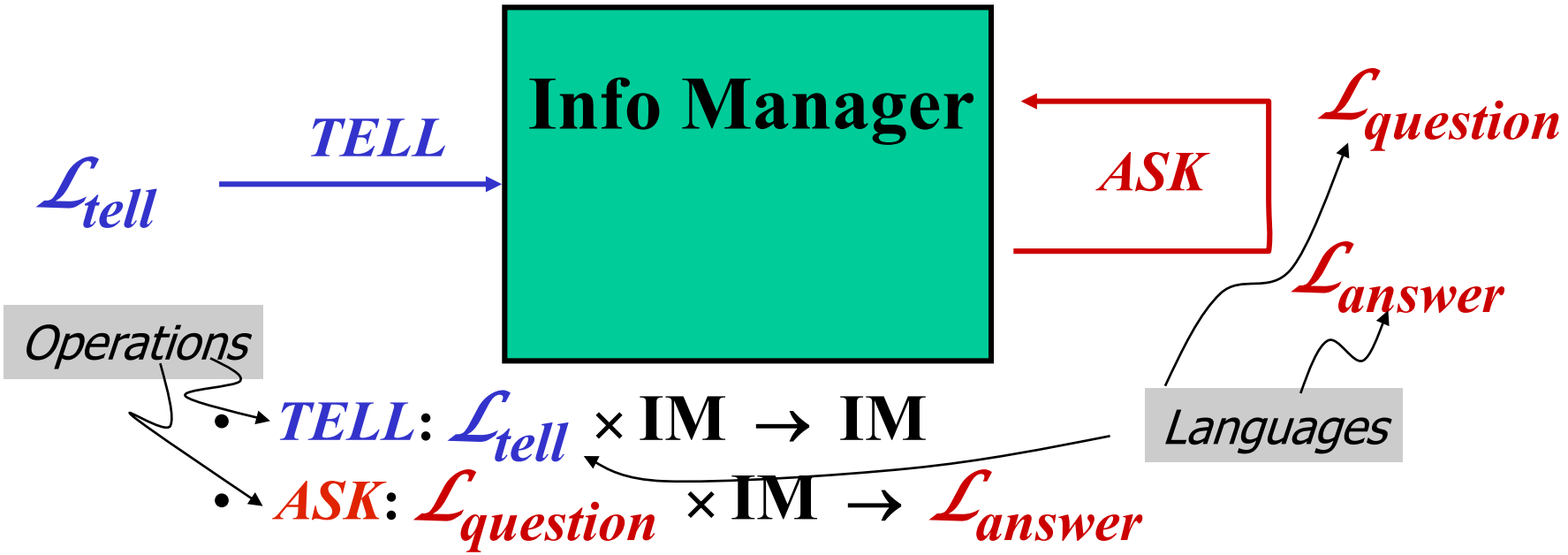
Methodologies for

- **building conceptual models**
- **designing relational schemas**
- **(integrating information sources)**

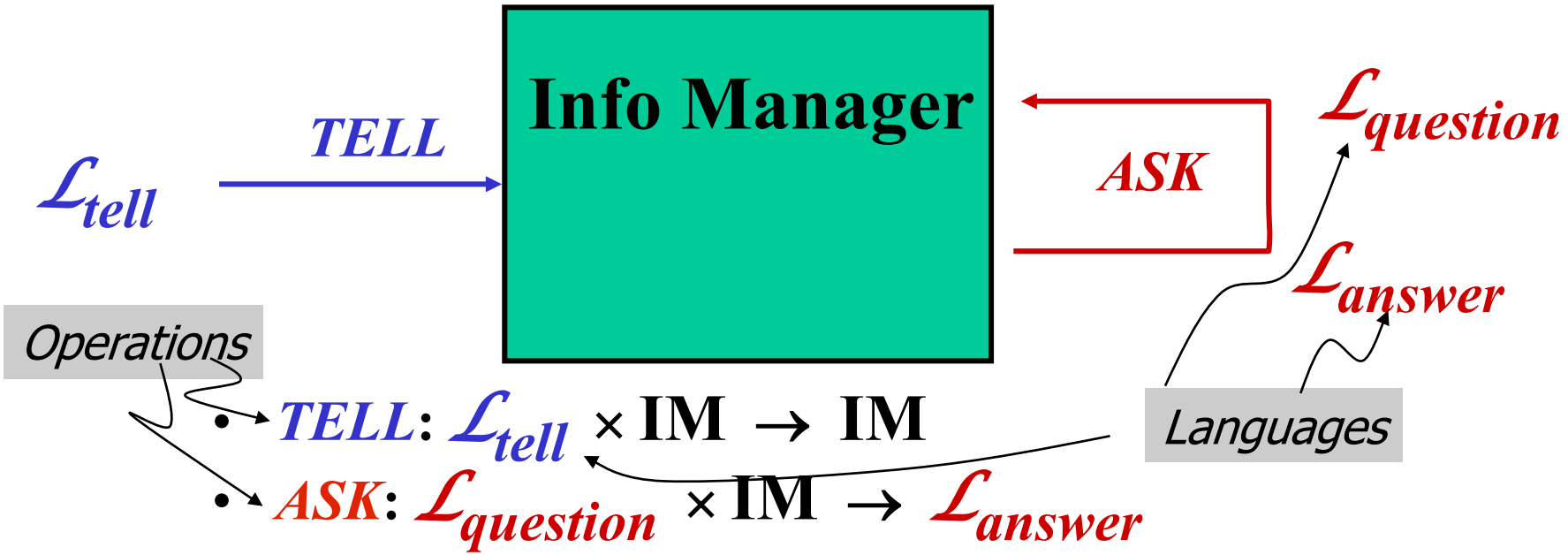
Functional view of Information Manager

[H. Levesque] - a unifying framework

Functional view of Information Manager
[H. Levesque] - a unifying framework



Functional view of Information Manager
[H. Levesque] - a unifying framework



Will discuss the

- ***Various languages, and how to use them***
- ***Specification of question answering***
- ***(Implementation of question answering – 198:437)***

"Symbol table"

an example of a very simple information manager

Want to keep track of all the 'words' encountered. (Sample application: *check for declared variables in a Java program*)

- tell the IM words
- ask it if some word has been encountered

Formal description of interface:

- $\mathcal{L}_{tell}$: words /* *give a grammar for 'word' ?!!* */
- $\mathcal{L}_{question}$: words
- $\mathcal{L}_{answer}$: { **yes, no** } for ***found/not found***

Specification of question answering: (based on sets)

IM ≡ set of words

TELL(w, IM) = {w } ∪ IM

ASK(w, IM) = if (**w ∈ IM**) then **yes** else **no**

Implementation: hash table, binary search tree,... (choice depends on relative frequency of TELL and ASKs);

Extensions

+ Clearing the table

- a second *TELL* operator, call it *MAKE_EMPTY*

$$MAKE\_EMPTY: \text{IM} \rightarrow \text{IM}$$

+ Ask for a count of occurrences of a word

- a second *ASK* operator, call it OCCURRENCE_COUNT

$$occur\_count: Words \times \text{IM} \rightarrow Integer$$

Specification: ***use multi-set , not simple set***

make_empty(IM)={}

occur_count(w,IM)=cardinality({ w | w in IM })

Implementation:

requires multi-set/bag implementation; so simple hash table not good enough

+ Some way to make the information *persistent*

- operations for saving and loading this symbol table
- more generally, naming multiple symbol tables, and operating on them

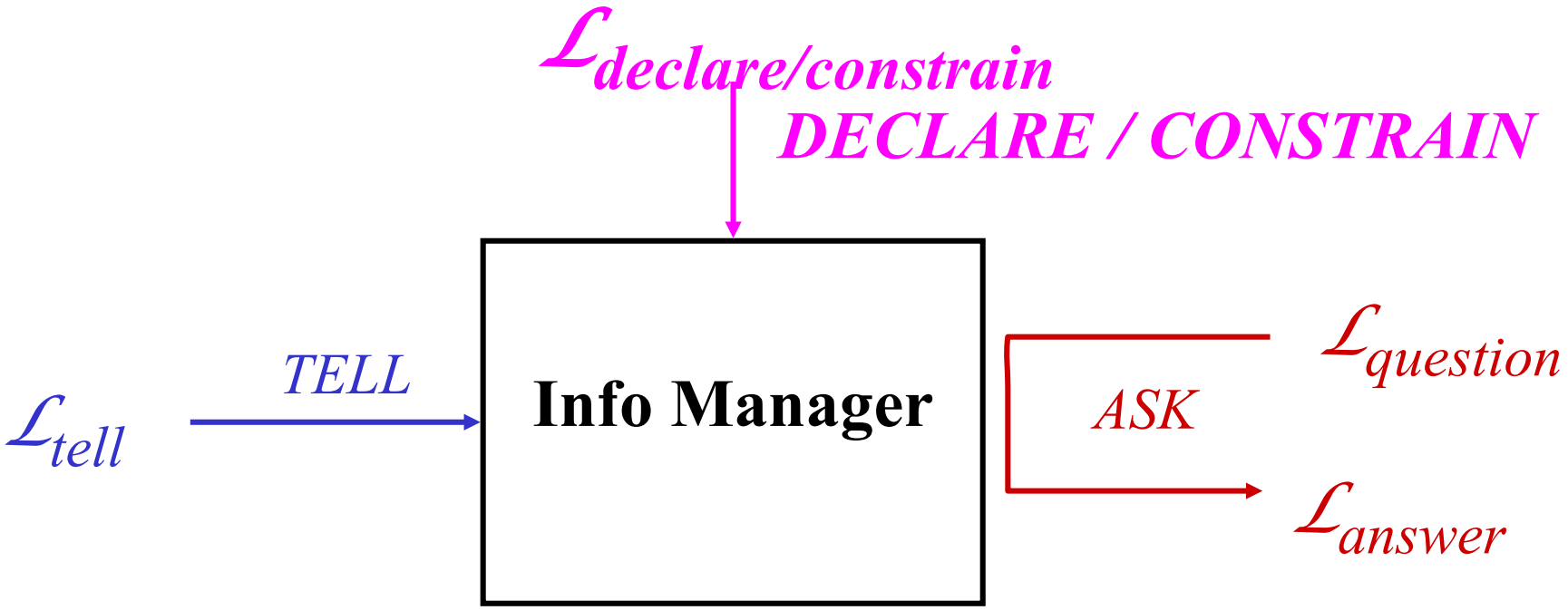
+ Maybe deal with concurrent access by multiple users/programs

Schema vs Data

In order to avoid ubiquitous data entry errors, and to speed up access, Information Managers *often* distinguish a subclass of Tell operations for "declaring" and "constraining" the kinds of facts that will be told. [NoSQL dbms claim not to do this.]

The structure of the data expected has the spirit of "type declarations" in programming languages.

Extended view of Information Manager



- ***DECLARE*** : $\mathcal{L}_{declare} \times \mathbf{IM} \rightarrow \mathbf{IM}$
- *TELL*: $\mathcal{L}_{tell} \times \mathrm{IM} \rightarrow \mathrm{IM} \cup$ ***Exceptions***
- *ASK*: $\mathcal{L}_{question} \times \mathrm{IM} \rightarrow \mathcal{L}_{answer}$

Desirable services provided by many IM

- **persistence** (maintain information even after program stops)
- **convenient access** (ability to ask questions "declaratively" rather than programmatically; hide and change implementation; queries optimized to speed up answering)
- **deal with massive amounts of facts told** (terabytes not uncommon; cannot be stored in main memory)
- **performance** (high speed even in the presence of many operations and much data)
- **maintain some notion of consistency in presence of multiple concurrent access**

Other features common to Database Management Systems (specialized IM):

- **resilience** (ability to survive hardware, software, power failures)
- **reliability** (almost always up – phone companies use dbms!)
- **scalability** (a recent phenomenon: data can grow incredibly fast: search engines cache the web)

Key people

- *IM implementer*
- *Schema designer*
- *Application developer*
- *IM administrator*