By Marina Barsky

Alternative data models:
1. RDF

Lecture 18

Strengths of relational model

- Logical data independence
- Ad hoc queries
- Mature technologies

Alternative data models

- The goal:
  - Explore alternative ways of data modeling
  - Overview of existing tools
  - Matching the tool to the task at hand

- Use cases that require alternative models
- Semantics of data models

Semantics – the study of meaning

We like mushrooms

Mushrooms scare Ann

- The same word has different meaning – different semantics
- We determine its semantics intuitively, based on our previous knowledge and the rules of the language
- We need to explicitly explain meaning to computer programs

Use case 1. Restaurant search web app

- Data about restaurants, their prices, addresses, cuisine, and open hours

Single table model: spreadsheets

Restaurant	Address	Cuisine	Price	Open
Deli Llama	Peachtree Rd	Deli	$	Mon, Tue, Wed, Thu, Fri
Peking Inn	Lake St	Chinese	$$$	Thu, Fri, Sat
Thai Tanic	Branch Dr	Thai	$$	Thu, Fri, Sat, Sun
Lord of the Fries	Flower Ave	Fast Food	$$	Tue, Wed, Thu, Fri, Sat, Sun
Wok This Way	Second St	Chinese	$	Mon, Tue, Wed, Thu, Fri, Sat, Sun
Award Wieners	Dorfold Mews	Fast Food	$	Mon, Tue, Wed, Thu, Fri, Sat

Semantics of a single table

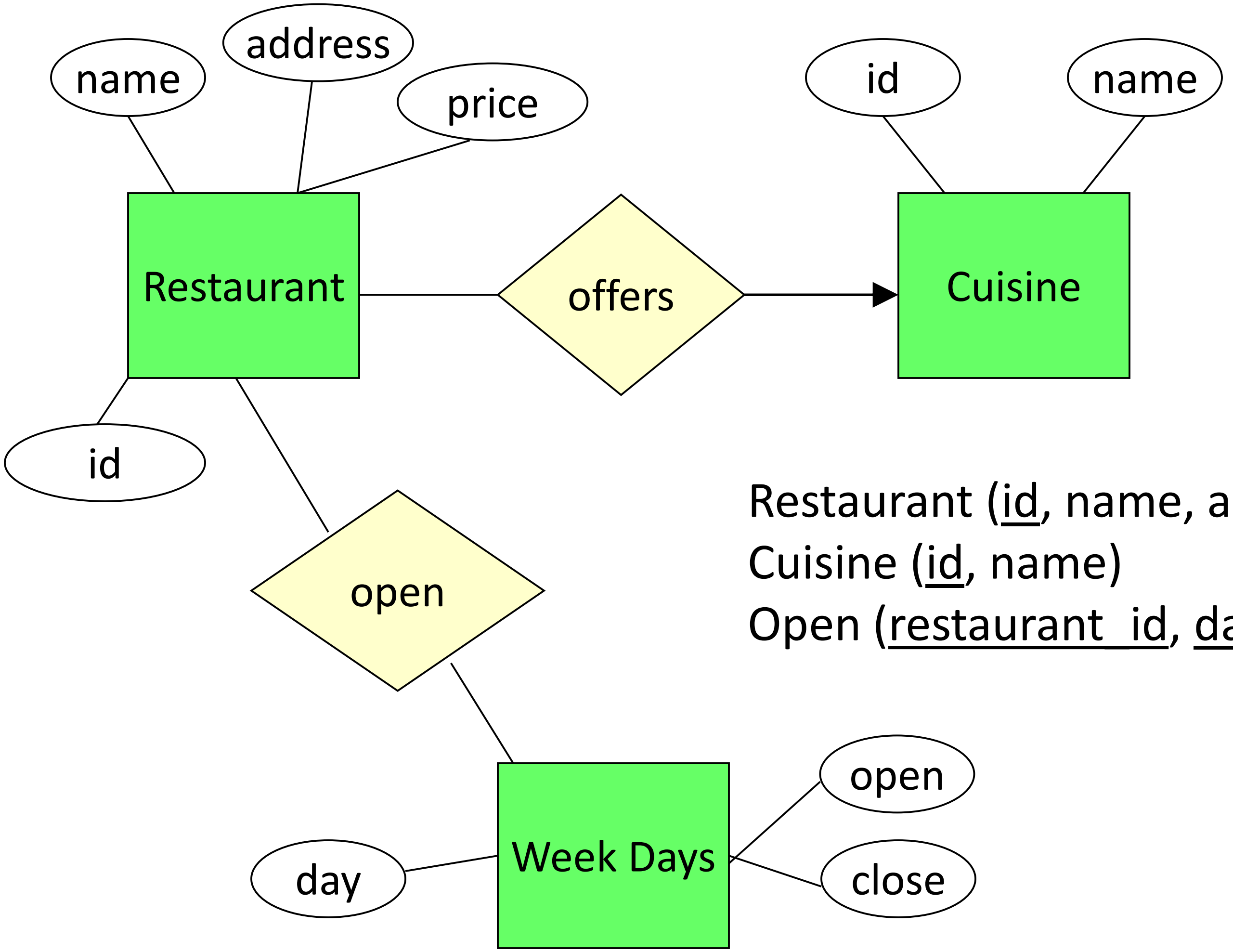
- The row and column explains what the value means to a person reading the data.
- The fact that Chinese is in the row Peking Inn and in the column Cuisine tells us that “Peking Inn serves Chinese food.”
- You know this because you understand what restaurants and cuisines are and because you’ve previously learned how to read a table.

Limitations of a single table model

- Multi-valued columns are not searchable
  - find the restaurants that will be open late on Friday night?

- Interconnected tables referencing the same data
  - a spreadsheet of our friends’ reviews of the restaurants - no easy way to search across both documents to find restaurants near our homes that our friends recommend

Relational model



Restaurant (id, name, address, price, cuisineID)
Cuisine (id, name)
Open (restaurant_id, day, open, close)

Relational model: sample tables

Restaurant				
id	Name	Address	CuisineID	Price
1	Deli Llama	Peachtree Rd	1	$
2	Peking Inn	Lake St	2	$$$

Hours			
Rest_id	Day	Open	Close
1	Mon	11	16
1	Tue	11	16
1	Wed	11	16
1	Thu	11	19
1	Fri	11	20
2	Thu	5	22
2	Fri	5	23
2	Sat	5	23

Cuisine	
id	Name
1	Deli
2	Chinese
3	Thai
4	Fast food

Benefits

- No redundancy.
- Ad hoc queries

Find all the restaurants that will be open at 10 p.m. on a Friday

```
SELECT Restaurant.Name, Cuisine.Name, Hours.Open, Hours.Close
FROM Restaurant, Cuisine, Hours
WHERE Restaurant.CuisineID=Cuisine.ID
AND Restaurant.ID=Hours.RestaurantID
AND Hours.Day="Fri"
AND Hours.Open<22
AND Hours.Close>22
```

Semantics of relational model

- The meaning of each value is described by the schema
- Each datum is labeled with what it means by the table in which it appears and by the column
- The model captures entities (restaurant, cuisine, week day) and relationships between them

- We convey this semantics to the computer program. We do not need to define what the restaurant is, but we can still get a list of restaurants with given properties

Extending scope of our search web app

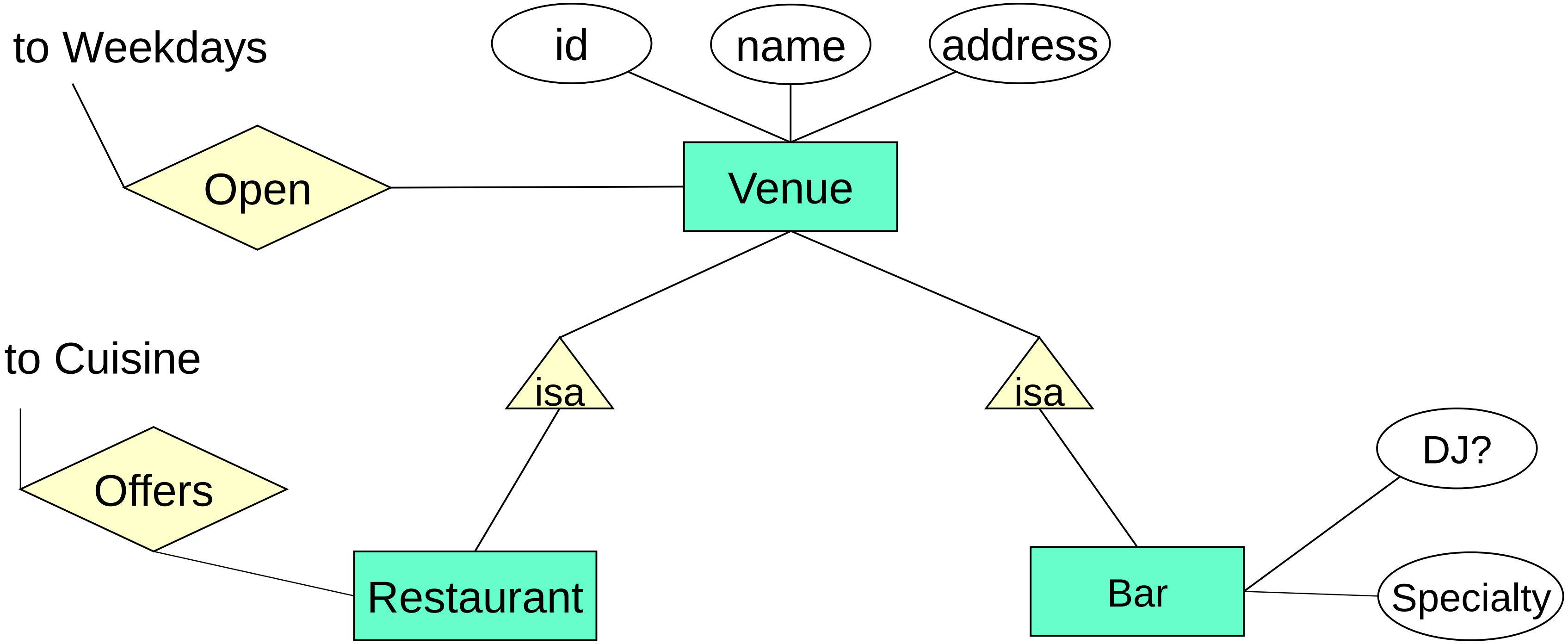
- Our restaurant search app is up and running
- We receive a new data to handle: bars

Bar			
Name	Address	**DJ**	**Specialty drink**
The bitter end	14th avenue	No	Beer
Peking Inn	Lake St	No	Scorpion Bowl
Hammer Time	Wildcat Dr	Yes	Hennessey

Integrating new data with existing model

- We cannot store bars in a separate table because restaurants and bars are related
    - Many restaurants serve as bars later in the evening
    - Bars and restaurants have common properties
    - Someone might want to query across both tables

Subclasses?



- Venue (<u>id</u>, name, address)
- Restaurant (<u>id</u>, cuisineID)
- Bar (<u>id</u>, DJ, specialty)

Relational model: problem 1 dealing with constantly evolving schema

- Relational databases are great for datasets where the data model is understood up front and there is a typical usage pattern
  - Product catalogs, contact lists, payroll systems

- Data integration across the Web
  - Rapidly changing types of data: venues could include a live music hall or a rental space for events
  - Cannot predict how data will be used

Changing the schema each time is expensive!

- Schema migration:
  - Load data from old tables into new tables
  - Update all triggers, functions and procedures
  - Update all queries and views
  - Update web site code

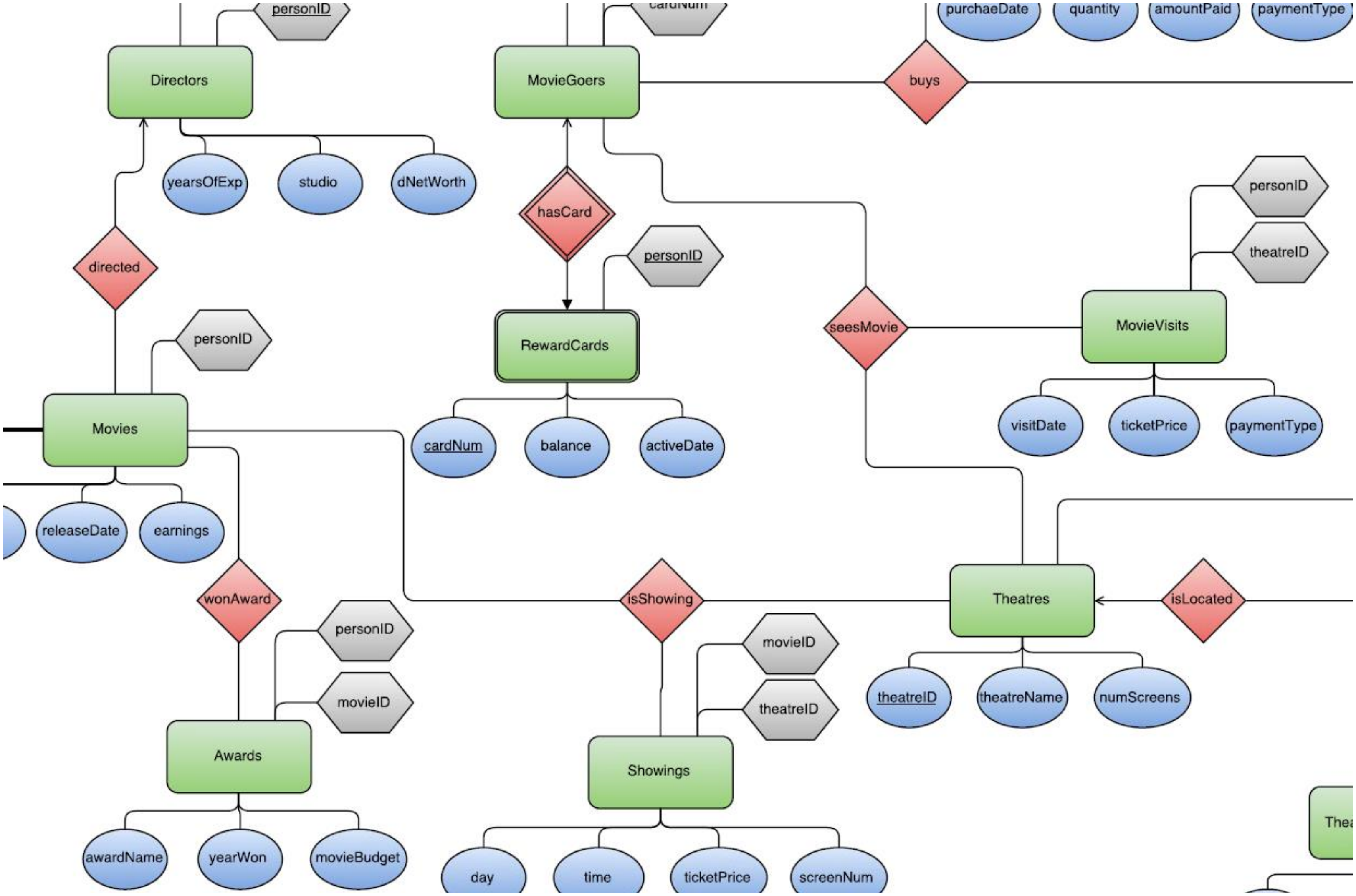
- Techniques for schema migration:
  - ORM (Hibernate)
  - Stored procedures
  - Complexities and bugs …
  - Downtime…

Relational model: problem 2
Very complex schemas

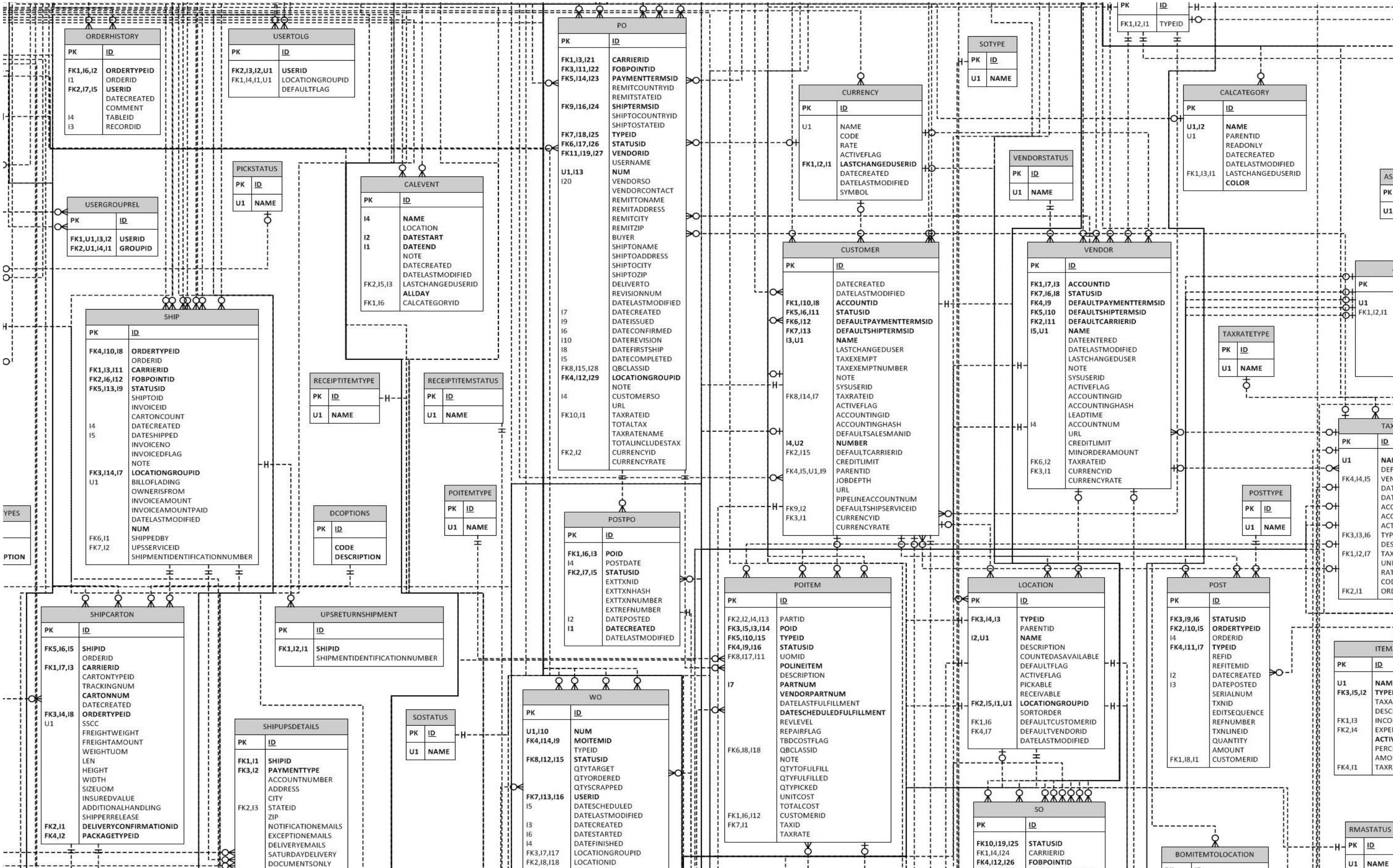
- Incredibly complicated schemas which include different data types
- Hundreds or thousands of inter-connected entities

- Challenge of using someone else’s relational data is understanding how the various tables relate to one another.
- This information—the data about the data representation—is called metadata
- Too frequently, data is archived, published, or shared without this critical metadata. Schemas need not become very large before metadata recovery becomes nearly impossible.

Movies and movie goers E/R



Hospital E/R (left upper corner)



ORDERHISTORY	
PK	ID
FK1,I6,I2	**ORDERTYPEID**
I1	ORDERID
FK2,I7,I5	**USERID**
	DATECREATED
	COMMENT
I4	TABLEID
I3	RECORDID

USERTOLG	
PK	ID
FK2,I3,I2,U1	**USERID**
FK1,I4,I1,U1	LOCATIONGROUPID
	DEFAULTFLAG

USERGROUPREL	
PK	ID
FK1,U1,I3,I2	**USERID**
FK2,U1,I4,I1	**GROUPID**

PICKSTATUS	
PK	ID
U1	**NAME**

CALEVENT	
PK	ID
I4	**NAME**
	LOCATION
I2	**DATESTART**
I1	**DATEEND**
	NOTE
	DATECREATED
	DATELASTMODIFIED
FK2,I5,I3	LASTCHANGEDUSERID
	ALLDAY
FK1,I6	CALCATEGORYID

SHIP	
PK	ID
FK4,I10,I8	**ORDERTYPEID**
	ORDERID
FK1,I3,I11	**CARRIERID**
FK2,I6,I12	**FOBPOINTID**
FK5,I13,I9	**STATUSID**
	SHIPTOID
	INVOICEID
	CARTONCOUNT
I4	DATECREATED
I5	DATESHIPPED
	INVOICENO
	INVOICEDFLAG
	NOTE
FK3,I14,I7	**LOCATIONGROUPID**
U1	BILLOFLADING
	OWNERISFROM
	INVOICEAMOUNT
	INVOICEAMOUNTPAID
	DATELASTMODIFIED
	NUM
FK6,I1	SHIPPEDBY
FK7,I2	UPSSERVICEID
	SHIPMENTIDENTIFICATIONNUMBER

YPES
PTION

RECEIPTITEMTYPE	
PK	ID
U1	**NAME**

RECEIPTITEMSTATUS	
PK	ID
U1	**NAME**

DCOPTIONS	
PK	ID
	CODE **DESCRIPTION**

POITEMTYPE	
PK	ID
U1	**NAME**

SHIPCARTON	
PK	ID
FK5,I6,I5	**SHIPID**
	ORDERID
FK1,I7,I3	**CARRIERID**
	CARTONTYPEID
	TRACKINGNUM
	CARTONNUM
	DATECREATED
FK3,I4,I8	**ORDERTYPEID**
U1	SSCC
	FREIGHTWEIGHT
	FREIGHTAMOUNT
	WEIGHTUOM
	LEN
	HEIGHT
	WIDTH
	SIZEUOM
	INSUREDVALUE
	ADDITIONALHANDLING
	SHIPPERRELEASE
FK2,I1	**DELIVERYCONFIRMATIONID**
FK4,I2	**PACKAGETYPEID**

UPSRETURNSHIPMENT	
PK	ID
FK1,I2,I1	**SHIPID**
	SHIPMENTIDENTIFICATIONNUMBER

SHIPUPSDETAILS	
PK	ID
FK1,I1	**SHIPID**
FK3,I2	**PAYMENTTYPE**
	ACCOUNTNUMBER
	ADDRESS
	CITY
FK2,I3	STATEID
	ZIP
	NOTIFICATIONEMAILS
	EXCEPTIONEMAILS
	DELIVERYEMAILS
	SATURDAYDELIVERY
	DOCUMENTSONLY

SOSTATUS	
PK	ID
U1	**NAME**

PO	
PK	ID
FK1,I3,I21	**CARRIERID**
FK3,I11,I22	**FOBPOINTID**
FK5,I14,I23	**PAYMENTTERMSID**
	REMITCOUNTRYID
	REMITSTATEID
FK9,I16,I24	**SHIPTERMSID**
	SHIPTOCOUNTRYID
	SHIPTOSTATEID
FK7,I18,I25	**TYPEID**
FK6,I17,I26	**STATUSID**
FK11,I19,I27	**VENDORID**
	USERNAME
U1,I13	**NUM**
I20	VENDORSO
	VENDORCONTACT
	REMITTONAME
	REMITADDRESS
	REMITCITY
	REMITZIP
	BUYER
	SHIPTONAME
	SHIPTOADDRESS
	SHIPTOCITY
	SHIPTOZIP
	DELIVERTO
	REVISIONNUM
	DATELASTMODIFIED
I7	DATECREATED
I9	DATEISSUED
I6	DATECONFIRMED
I10	DATEREVISION
I8	DATEFIRSTSHIP
I5	DATECOMPLETED
FK8,I15,I28	QBCLASSID
FK4,I12,I29	**LOCATIONGROUPID**
	NOTE
I4	CUSTOMERSO
	URL
FK10,I1	TAXRATEID
	TOTALTAX
	TAXRATENAME
	TOTALINCLUDESTAX
FK2,I2	CURRENCYID
	CURRENCYRATE

POSTPO	
PK	ID
FK1,I6,I3	**POID**
I4	POSTDATE
FK2,I7,I5	**STATUSID**
	EXTTXNID
	EXTTXNHASH
	EXTTXNNUMBER
	EXTREFNUMBER
I2	DATEPOSTED
I1	**DATECREATED**
	DATELASTMODIFIED

WO	
PK	ID
U1,I10	**NUM**
FK4,I14,I9	**MOITEMID**
	TYPEID
FK8,I12,I15	**STATUSID**
	QTYTARGET
	QTYORDERED
	QTYSCRAPPED
FK7,I13,I16	**USERID**
I5	DATESCHEDULED
	DATELASTMODIFIED
I3	DATECREATED
I6	DATESTARTED
I4	DATEFINISHED
FK3,I7,I17	LOCATIONGROUPID
FK2,I8,I18	LOCATIONID

CURRENCY	
PK	ID
U1	NAME
	CODE
	RATE
	ACTIVEFLAG
FK1,I2,I1	**LASTCHANGEDUSERID**
	DATECREATED
	DATELASTMODIFIED
	SYMBOL

CUSTOMER	
PK	ID
	DATECREATED
	DATELASTMODIFIED
FK1,I10,I8	**ACCOUNTID**
FK5,I6,I11	**STATUSID**
FK6,I12	**DEFAULTPAYMENTTERMSID**
FK7,I13	**DEFAULTSHIPTERMSID**
I3,U1	**NAME**
	LASTCHANGEDUSER
	TAXEXEMPT
	TAXEXEMPTNUMBER
	NOTE
	SYSUSERID
FK8,I14,I7	TAXRATEID
	ACTIVEFLAG
	ACCOUNTINGID
	ACCOUNTINGHASH
	DEFAULTSALESMANID
I4,U2	**NUMBER**
FK2,I15	DEFAULTCARRIERID
	CREDITLIMIT
FK4,I5,U1,I9	PARENTID
	JOBDEPTH
	URL
	PIPELINEACCOUNTNUM
FK9,I2	DEFAULTSHIPSERVICEID
FK3,I1	CURRENCYID
	CURRENCYRATE

POITEM	
PK	ID
FK2,I2,I4,I13	PARTID
FK3,I5,I3,I14	**POID**
FK5,I10,I15	**TYPEID**
FK4,I9,I16	**STATUSID**
FK8,I17,I11	UOMID
	POLINEITEM
	DESCRIPTION
I7	**PARTNUM**
	VENDORPARTNUM
	DATELASTFULFILLMENT
	DATESCHEDULEDFULFILLMENT
	REVLEVEL
	REPAIRFLAG
	TBDCOSTFLAG
FK6,I8,I18	QBCLASSID
	NOTE
	QTYTOFULFILL
	QTYFULFILLED
	QTYPICKED
	UNITCOST
	TOTALCOST
FK1,I6,I12	CUSTOMERID
FK7,I1	TAXID
	TAXRATE

SOTYPE	
PK	ID
U1	**NAME**

PK	ID
FK1,I2,I1	TYPEID

VENDORSTATUS	
PK	ID
U1	**NAME**

VENDOR	
PK	ID
FK1,I7,I3	**ACCOUNTID**
FK7,I6,I8	**STATUSID**
FK4,I9	**DEFAULTPAYMENTTERMSID**
FK5,I10	**DEFAULTSHIPTERMSID**
FK2,I11	**DEFAULTCARRIERID**
I5,U1	**NAME**
	DATEENTERED
	DATELASTMODIFIED
	LASTCHANGEDUSER
	NOTE
	SYSUSERID
	ACTIVEFLAG
	ACCOUNTINGID
	ACCOUNTINGHASH
	LEADTIME
I4	ACCOUNTNUM
	URL
	CREDITLIMIT
	MINORDERAMOUNT
FK6,I2	TAXRATEID
FK3,I1	CURRENCYID
	CURRENCYRATE

LOCATION	
PK	ID
FK3,I4,I3	**TYPEID**
	PARENTID
I2,U1	**NAME**
	DESCRIPTION
	COUNTEDASAVAILABLE
	DEFAULTFLAG
	ACTIVEFLAG
	PICKABLE
	RECEIVABLE
FK2,I5,I1,U1	**LOCATIONGROUPID**
	SORTORDER
FK1,I6	DEFAULTCUSTOMERID
FK4,I7	DEFAULTVENDORID
	DATELASTMODIFIED

SO	
PK	ID
FK10,I19,I25	**STATUSID**
FK1,I4,I24	CARRIERID
FK4,I12,I26	**FOBPOINTID**

CALCATEGORY	
PK	ID
U1,I2	**NAME**
U1	PARENTID
	READONLY
	DATECREATED
	DATELASTMODIFIED
FK1,I3,I1	LASTCHANGEDUSERID
	COLOR

TAXRATETYPE	
PK	ID
U1	**NAME**

POSTTYPE	
PK	ID
U1	**NAME**

POST	
PK	ID
FK3,I9,I6	**STATUSID**
FK2,I10,I5	**ORDERTYPEID**
I4	ORDERID
FK4,I11,I7	**TYPEID**
	REFID
	REFITEMID
I2	DATECREATED
I3	DATEPOSTED
	SERIALNUM
	TXNID
	EDITSEQUENCE
	REFNUMBER
	TXNLINEID
	QUANTITY
	AMOUNT
FK1,I8,I1	CUSTOMERID

BOMITEMTOLOCATION

AS	
PK	
U1	

PK	
U1	
FK1,I2,I1	

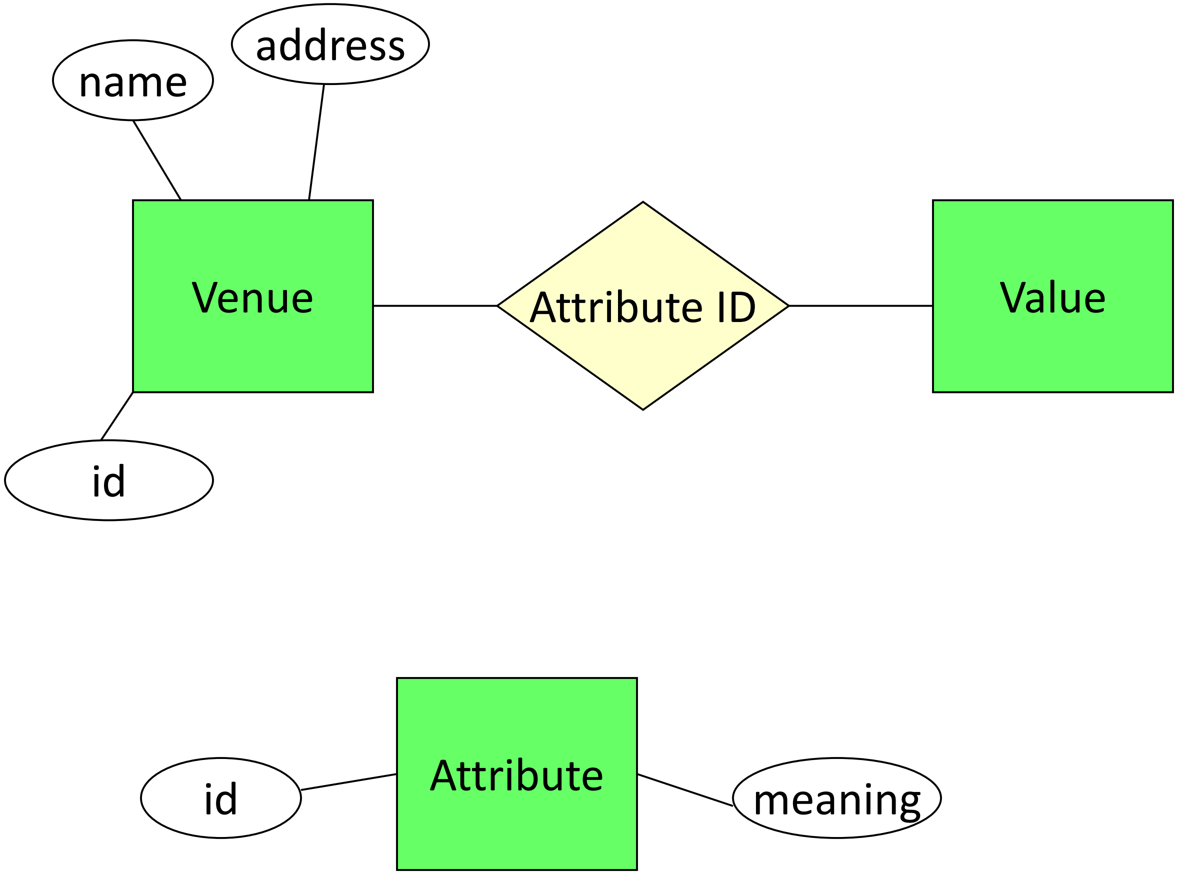
TAX	
PK	ID
U1	**NA**
	DEF
FK4,I4,I5	VEN
	DAT
	DAT
	ACC
	ACC
	ACT
FK3,I3,I6	TYP
	DES
FK1,I2,I7	TAX
	UNI
	RAT
	COD
FK2,I1	ORD

ITEM	
PK	ID
U1	**NAM**
FK3,I5,I2	**TYPE**
	TAXA
	DESC
FK1,I3	INCO
FK2,I4	EXPE
	ACTI
	PERC
	AMO
FK4,I1	TAXR

RMASTATUS	
PK	ID
U1	**NAME**

Designing more flexible model for web data integration

Making it extendable from the beginning



Flexible schema

Venue		
id	Name	Address
1	Deli Llama	Peachtree Rd
2	Peking Inn	Lake St
3	Thai Tanic	Branch Dr

Properties		
VenueID	Attribute ID	Value
1	1	Deli
1	2	$
2	1	Chinese
2	2	$$$
2	3	Scorpion Bowl
2	4	No

Attributes	
id	Meaning
1	Cuisine
2	Price
3	Specialty
4	DJ

Flexible schema: adding concert venues

Venue		
id	Name	Address
1	Deli Llama	Peachtree Rd
2	Peking Inn	Lake St
3	Thai Tanic	Branch Dr

Attributes	
id	Meaning
1	Cuisine
2	Price
3	Specialty
4	DJ
5	Live Music
6	Music Genre

Properties		
VenueID	Attribute ID	Value
1	1	Deli
1	2	$
2	1	Chinese
2	2	$$$
2	3	Scorpion Bowl
2	4	No
3	5	Yes
3	6	Jazz

Converting everything into attribute-value pairs

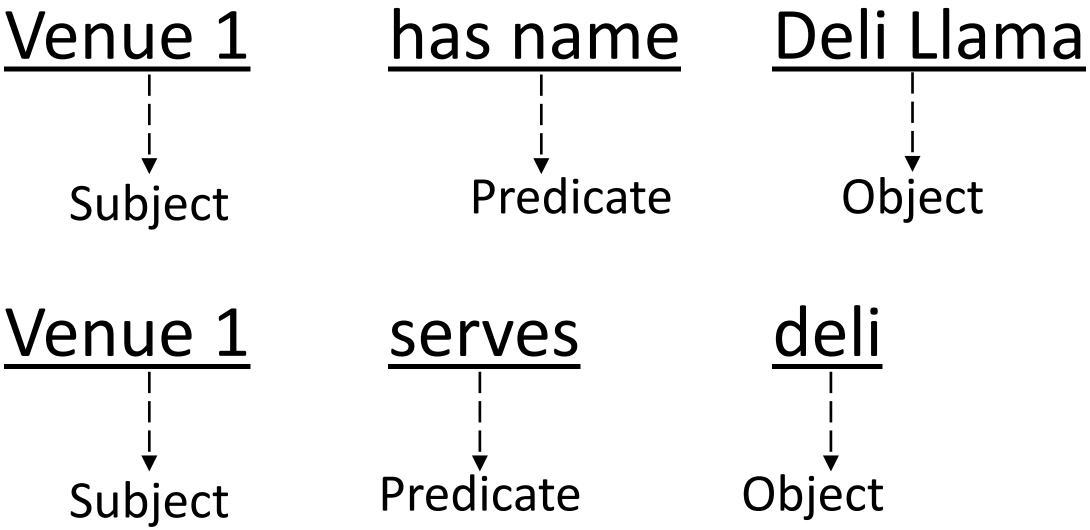
Attributes	
id	Meaning
1	Cuisine
2	Price
3	Specialty
4	DJ
5	Live Music
6	Music Genre
7	Name
8	Address

Properties		
VenueID	Attribute ID	Value
1	1	Deli
1	2	$
2	1	Chinese
2	2	$$$
2	3	Scorpion Bowl
2	4	No
1	7	Deli Llama
2	7	Peking Inn
1	8	Peachtree Road
2	8	Lake ST

Joining everything into a single table

Venues		
VenueID	Attribute	Value
1	Cuisine	Deli
1	Price	$
2	Cuisine	Chinese
2	Price	$$$
2	Specialty	Scorpion Bowl
2	DJ	No
1	Name	Deli Llama
2	Name	Peking Inn
1	Address	Peachtree Road
2	Address	Lake ST

Semantic meaning



Venue 1 has name Deli Llama

Subject Predicate Object

Venue 1 serves deli

Subject Predicate Object

- Fully parameterized venue table - represent arbitrary facts about food and music venues
- This three-column format is known as a **triple**
- Each triple is composed of a subject, a predicate, and an object.
- Each triple represents simple linguistic statements

Semantic table

Venues		
Subject	Predicate	Object
S1	Cuisine	Deli
S1	Price	$
S2	Cuisine	Chinese
S2	Price	$$$
S2	Specialty	Scorpion Bowl
S2	DJ	No
S1	Name	Deli Llama
S2	Name	Peking Inn
S1	Address	Peachtree Road
S2	Address	Lake ST

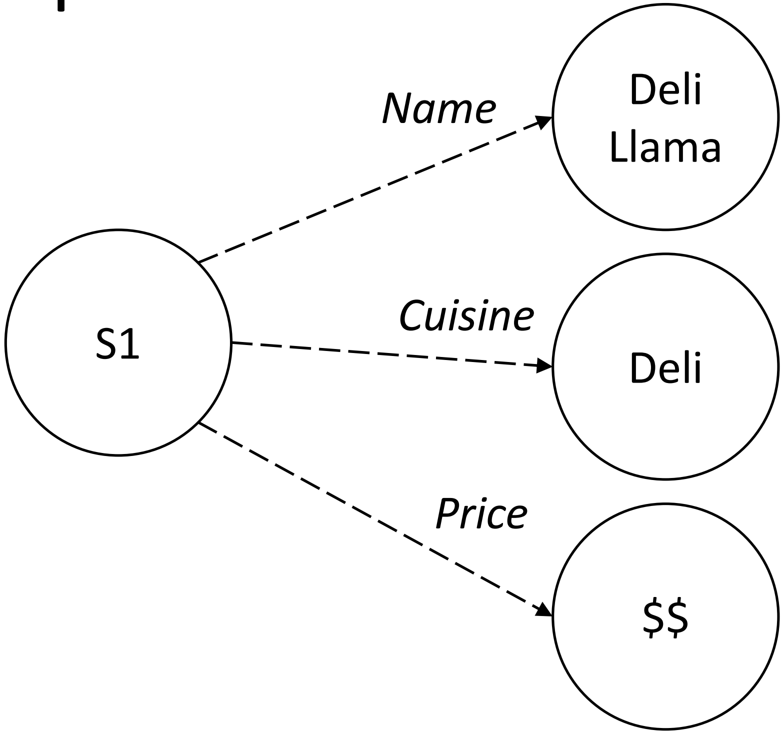
Semantic modeling

- The **subject** in a triple corresponds to an entity—a “thing” for which we have a conceptual class.
  - People, places, even periods of time and ideas.
- **Predicates** are a property of the entity to which they are attached.
  - A person’s name or birth date or a business’s mailing address are all examples of predicates.
- **Objects** fall into two classes:
  - entities that can be the subject in other triples
  - literal values such as strings or numbers.

Data graph

- Multiple triples can be tied together by using the same subjects and objects in different triples
- As we assemble these chains of relationships, they form a *directed labeled graph*

Graph of venues: sample node

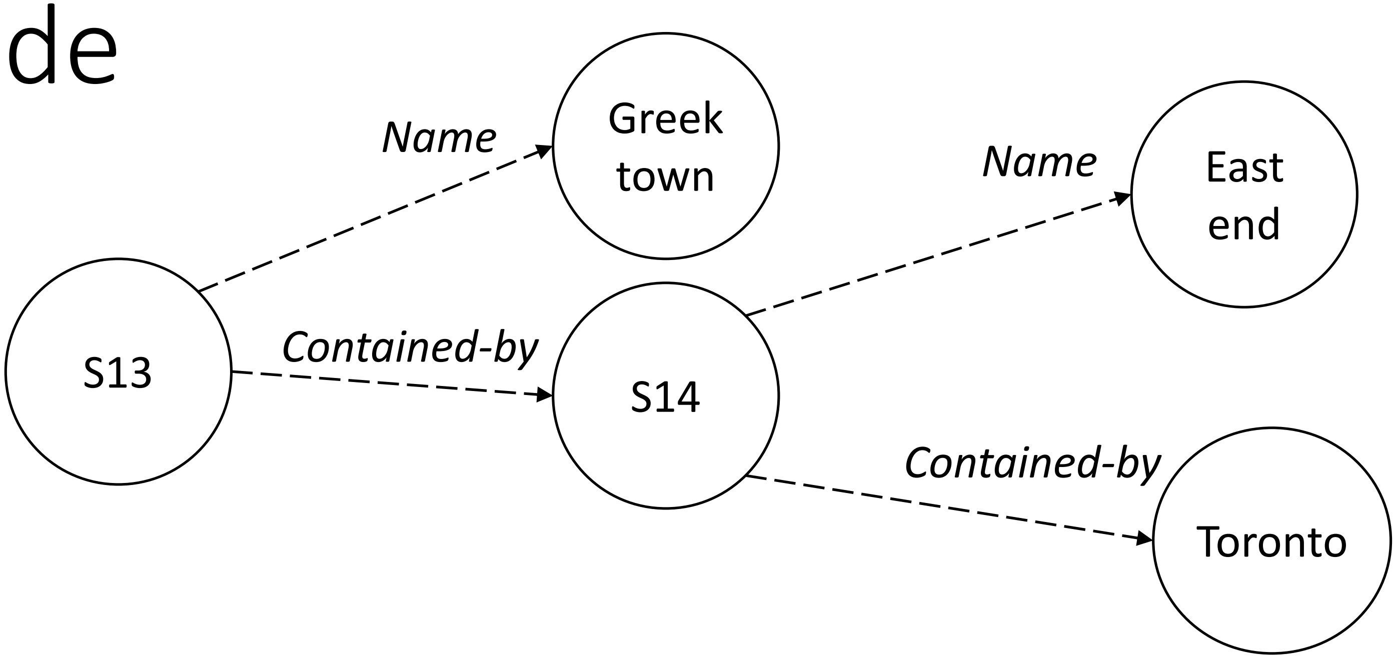


Integrating new entity: neighborhood

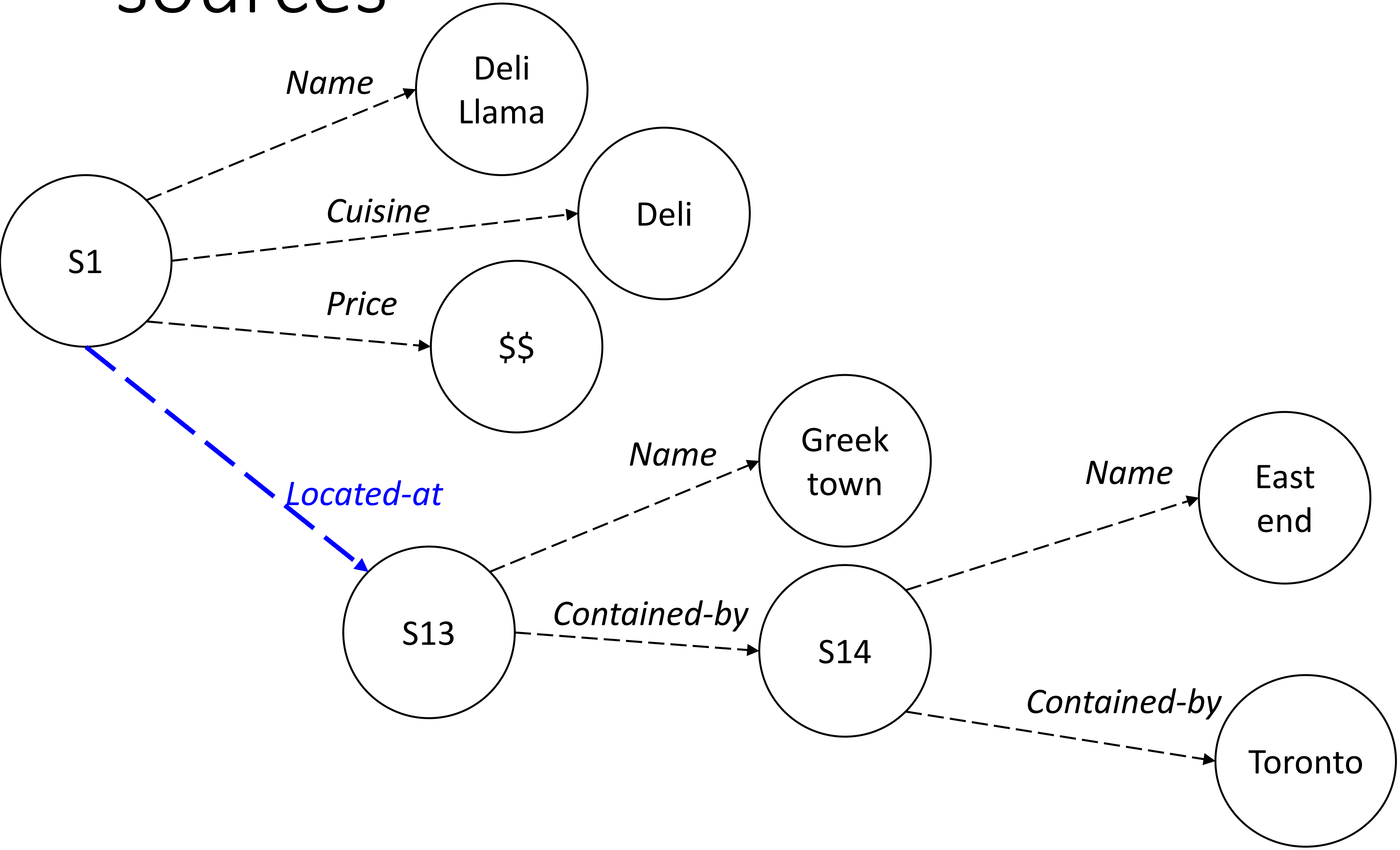
Neighborhoods		
Subject	Predicate	Object
S11	Name	Financial District
S11	Contained-by	S12
S12	Name	Downtown core
S12	Contained-by	Toronto
S13	Name	Greektown
S13	Contained-by	S14
S14	Name	East end
S14	Contained-by	Toronto

- Let’s relax the meaning of the first column and assert that subjects can represent any entity
- We can then append neighborhood information to the same table as our restaurant data

Graph of neighborhoods: sample node



Integrating data from multiple sources



Advantages of semantic model 1/5

- Very flexible – we can add any new data type into the same table
    - Espresso machine locations, coffee shops, book stores, gas stations …

Advantages of semantic model 2/5

- Self-describing data – do not need a special schema definition
  - the semantic relationships that previously were inferred from the table and column are contained in data itself

Advantages of semantic model 3/5

- Easy integration of data from multiple sources
    - Just add new data to the same table and create a link to the old data if needed

Advantages of semantic model 4/5

- We can add new features without affecting legacy software – no schema migration, there is the same simple schema all the time

Advantages of semantic model 5/5

- Simple common data interface:
  - everyone can write an app in Python, Ruby or Ruby to plot crime statistics on the map or find cuisines in the walking distance from the movie

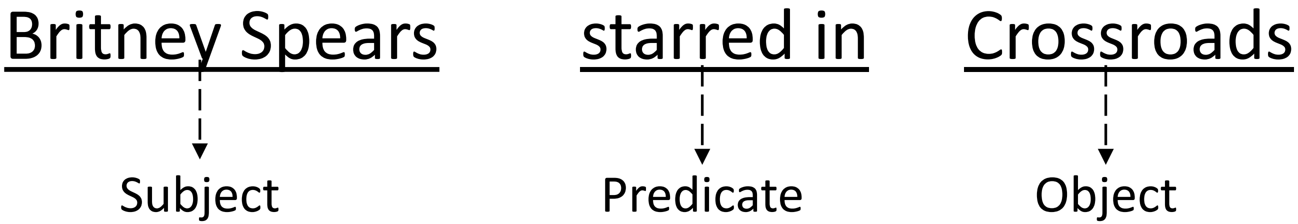
Semantic web

- RDF (Resource Description Framework) web data can be thought of in terms of a decentralized directed labeled graph wherein the arcs start with subject URIs, are labeled with predicate URIs, and end up pointing to object URIs or scalar values
- Uniform Resource Identifier (URI) is a string of characters used to identify a resource (for example for books - urn:isbn:0-486-27557-4)

Example: Celebrities dataset

- Entities – celebrity, relationship, rehab, album, movie
- Entities can be both subject and object
- Predicates:
  - end
  - enemy
  - person
  - released_album
  - starred_in
  - start
  - with

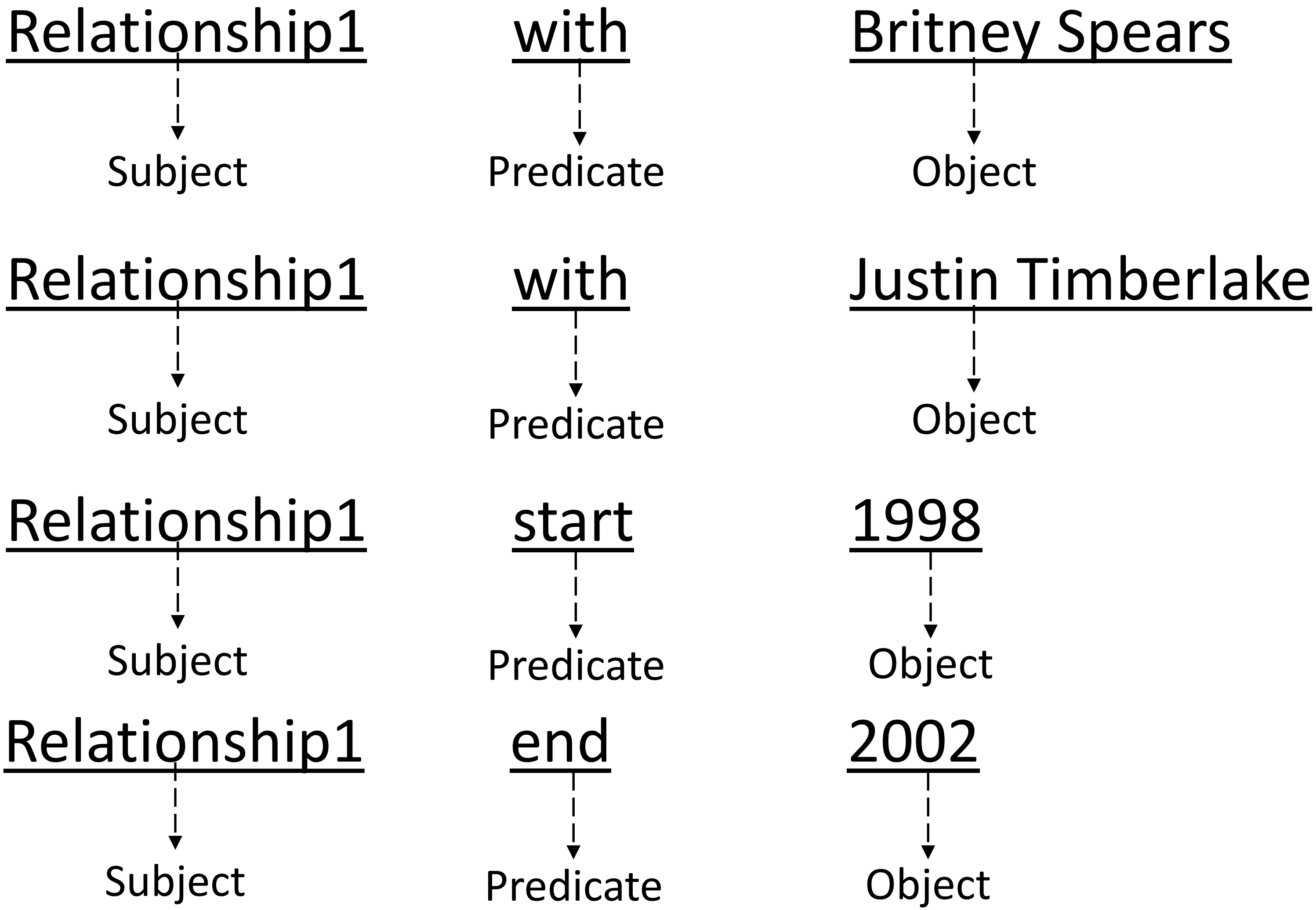
Let’s model celebrity



Britney Spears starred in Crossroads

Subject Predicate Object

Let’s model relationships



Relationship1 with Britney Spears

Subject Predicate Object

Relationship1 with Justin Timberlake

Subject Predicate Object

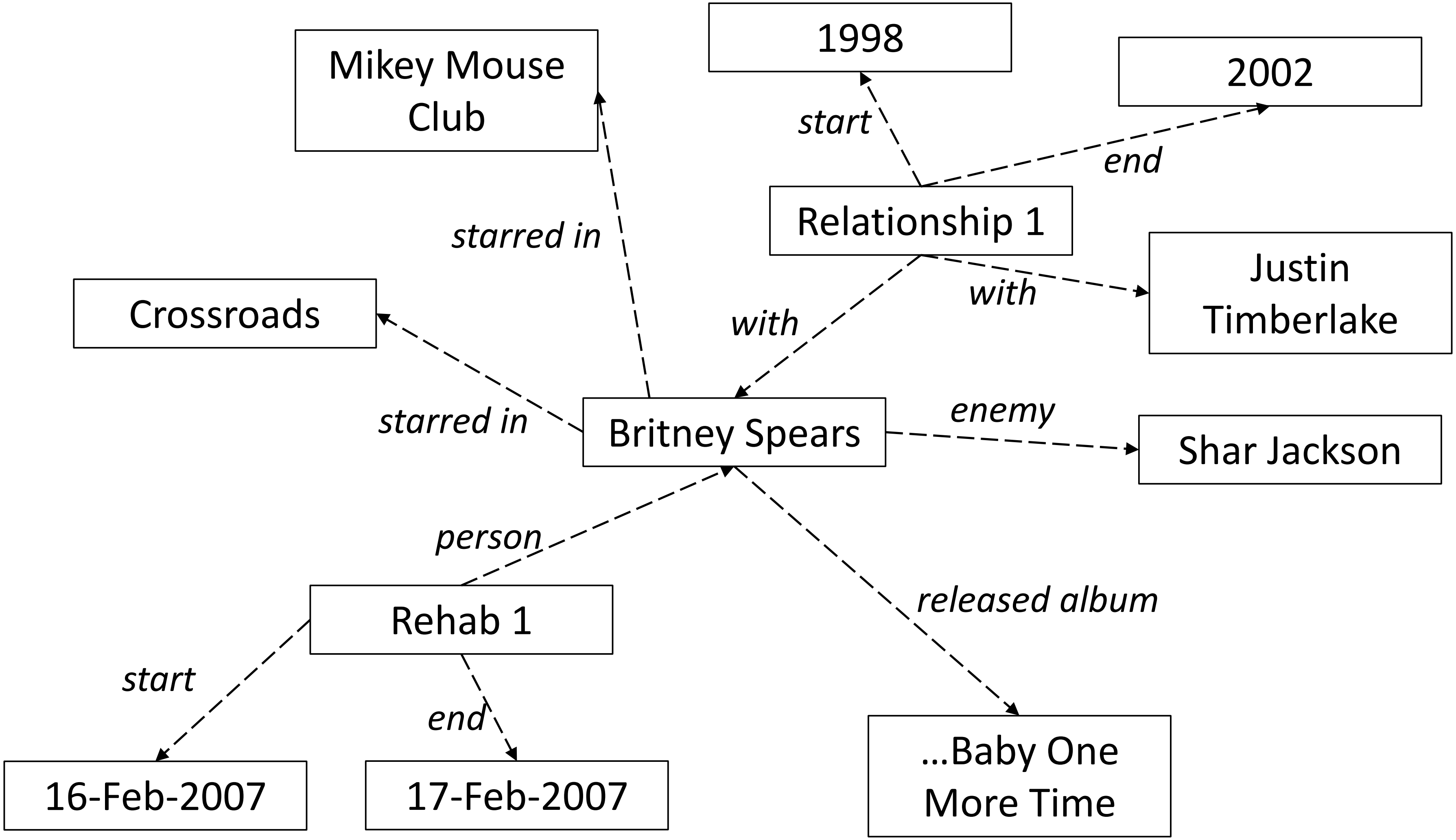
Relationship1 start 1998

Subject Predicate Object

Relationship1 end 2002

Subject Predicate Object

Celebrity graph: sample node



Example 1. Which celebrities have dated more than one star?

```
CREATE VIEW movie_stars AS
SELECT distinct subject FROM celebrities
WHERE predicate = 'starred_in';

CREATE VIEW relationships AS
SELECT distinct R1.object AS celeb1, R2.object AS celeb2
FROM celebrities R1, celebrities R2
WHERE R1.predicate = 'with' AND R2.predicate = 'with'
AND R1.subject = R2.subject AND R1.object < R2.object;

SELECT distinct celeb1, COUNT(celeb2) AS cnt FROM relationships
WHERE celeb2 IN (SELECT * FROM movie_stars )
GROUP BY celeb1
HAVING cnt >=2;
```

Example 2. Which musicians have spent time in rehab?

```
CREATE VIEW musicians
AS select distinct subject from celebrities
where predicate = 'released_album';

 CREATE VIEW rehab_celebs
 AS SELECT distinct object FROM  celebrities
 WHERE predicate = 'person';

SELECT * from musicians INTERSECT SELECT * from
rehab_celebs;
```

Triplestores implementation: index

- A common technique: cross-indexing the subject, predicate, and object in all different permutations (ops, osp, pos, pso, sop, spo) so that all triple queries can be answered through fast lookups
- Each of the indexes holds a different permutation of each triple that is stored in the graph. The name of the index indicates the ordering of the terms in the index (i.e., the pos index stores the predicate, then the object, and then the subject, in that order).

Triplestores implementation: query format

- The basic query method takes a (subject, predicate, object) pattern and returns all triples that match the pattern.
- Terms in the triple that are set to None are treated as wildcards.
- The triples method determines which index to use based on which terms of the triple are wildcarded, and then iterates over the appropriate index,

Queries can be implemented as triple matchings

(*, ‘with’, ‘Britney Spears’)

- We can put the results into a list variable – relationships

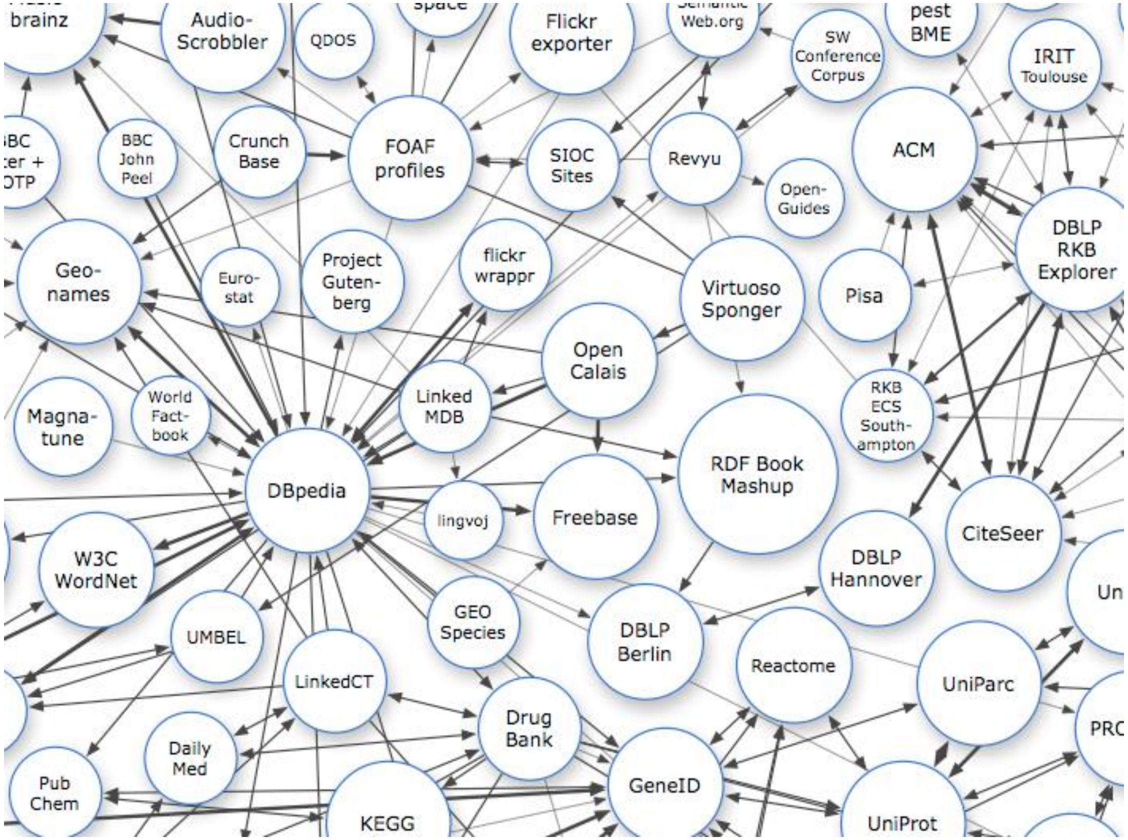
(‘?relationships’, ‘with’, ‘Britney Spears’)

- And use the results in a subsequent queries:

(‘relationships’, ‘with’, ‘?partners’)

http://linkeddata.org/

The goal: exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF



NoSQL ("Not only SQL") databases

NoSQL database systems

- New generation of non-relational database systems

- Properties:
  - Flexibility: schema-less
  - Scalability: inherently parallelizable

Main types of NoSQL systems

- Key-value databases: key-value pairs

  Redis, SimpleDB

- Document databases: key-value stores where values are entire documents

  CouchDB, MongoDB

- Wide-column databases: multi-dimensional sorted map

  Google's BigTable, Cassandra

- Graph databases: store data as connected nodes of a graph

  HyperGraphDB, multiple implementations of semantic RDF triplestores