Introduction to Graph Data Management

Claudio Gutierrez

Center for Semantic Web Research (CIWS)
Department of Computer Science
Universidad de Chile

EDBT Summer School – Palamos 2015
Joint Work With

Renzo Angles
Universidad de Talca, Chile
To the memory of Alberto Mendelzon, one of the first explorers of graph databases
1. Basic notions and vocabulary
2. Graph Database Models
3. Classical models
4. Some questions
Database perspective

Logic + Ontology vocabulary
(Concepts + knowledge)
∀x ∃y(R(x, y) → ∃zQ(z))
A ∪ B, C → D ∩ E

data + schema
(entities + relations)

Syntax + formats
(Text + Links + code languages)

KR - Logic

Databases

Information Retrieval

Trust

Proof

Digital Signature

Unicode

URI

C. Gutierrez – EDBT Summer School - Palamos 2015
Manage huge volumes of data with logical precision
Separate modeling from implementation levels
Database Approach

- Manage huge volumes of data with logical precision
- Separate modeling from implementation levels

As opposed to AI: DB primary concern is scalability. Then expressive power
The Database Approach

- Manage huge volumes of data with logical precision
- Separate modeling from implementation levels

As opposed to AI: **DB primary concern is scalability.** Then expressive power
As opposed to IR: **DB primary concern is precision.** Then scalability (recall).
Database Technology

Data Structure: Graphs

Query languages

Native Data Store

Files

RDBMS

APIs

Applications

Services

Database Technology

C. Gutierrez – EDBT Summer School - Palamos 2015
This Talk: Database Modeling Level

Data Structure: Graphs

Query languages

Native Data Store

Files

RDBMS

APIs

Applications

Services

MySQL

MSQL

Oracle

DB2

Postgres

This Talk: Database Modeling Level

C. Gutierrez – EDBT Summer School - Palamos 2015
Database Models: Codd’s definition

- Data structures
- Integrity constraints
- Query Language
Data structure

*Data and/or the schema are represented by graphs, or by data structures generalizing the notion of graph (hypergraphs or hypernodes).*
Integrity constraints

*Integrity constraints* enforce data consistency. These constraints can be grouped in schema-instance consistency, identity and referential integrity, and functional and inclusion dependencies.
Query Language

*Data manipulation* is expressed by graph transformations, or by operations whose main primitives are on graph features like paths, neighborhoods, subgraphs, graph patterns, connectivity, and graph statistics.
Evolution of Classical Database Models

- Mathematical Logic
- Knowledge Representation
- Graph Theory

Hierarchical
Network
Logic Programming
Deductive
Statistical Databases

Relational
Semantic

Object Oriented Programming
Object Oriented

Multidimensional

Graph

Semistructured
XML

Theoretical Basis
Database Model
Influence
Golden Age of Graph Databases
The Y2K change of phase

Hardware able to process big graphs

1. As always, software behind hardware, and theory/models behind software
2. We are just understanding the problems: thus keep your 6 senses open to understand where the field is heading to
3. No simple solutions; be humble
Some minimalistic tasks and questions

1. Use cases
2. Data structure(s)
3. Query language(s) / basic operators
4. Benchmarks
5. Open versus closed world
6. Centralized or distributed?
7. Dynamics (transaction $\rightarrow$ interaction?)
## Use cases: Social Networks domain

<table>
<thead>
<tr>
<th>Chapter title</th>
<th>Use Case (local)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Looking for Social Structure</td>
<td>+ Directed to undirected binary relations + Remove relations</td>
</tr>
<tr>
<td>Attributes and Relations</td>
<td>+ Extract a subnetwork based on attributes + Group actors based on attributes</td>
</tr>
<tr>
<td></td>
<td>+ Selective grouping of actors based on attributes</td>
</tr>
<tr>
<td>Cohesive Subgroups</td>
<td>+ Extract the subnetwork induced by cliques of size n + Build a hierarchy of cliques</td>
</tr>
<tr>
<td>Frienship</td>
<td>+ Extract subnetwork by time</td>
</tr>
<tr>
<td>Affiliations</td>
<td>+ Two-mode network to one-mode network</td>
</tr>
<tr>
<td>Center and Periphery</td>
<td>+ Group multiple binary relations</td>
</tr>
<tr>
<td>Brokers and Bridges</td>
<td>+ Extract egonetwork of an actor + Remove relations between groups</td>
</tr>
<tr>
<td>Diffusion</td>
<td>+ Selective counting of neighbors + Operations between attributes + Change relation direction based on attributes</td>
</tr>
<tr>
<td>Prestige</td>
<td>+ Discretize an attribute</td>
</tr>
<tr>
<td>Ranking</td>
<td>+ Find triads by type</td>
</tr>
<tr>
<td>Genealogies and Citations</td>
<td>+ Loop removal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subgraph family</th>
<th>Use Case (Global)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paths and Cycles</td>
<td>+ Geodesics</td>
</tr>
<tr>
<td>Groups (k-neighbors, k-core, n-cliques, k-plex, etc.)</td>
<td>+ Detect cohesive subgroups + Egonetworks + Input Domain</td>
</tr>
<tr>
<td>Connected components</td>
<td>+ Connected components + Clustering + Bicomponents and brockers</td>
</tr>
</tbody>
</table>

C. Gutierrez – EDBT Summer School - Palamos 2015
Use cases: Biology domain

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Graph Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chemical structure associated with a node</td>
<td>Node matching</td>
</tr>
<tr>
<td>Find the difference in metabolisms between two microbes</td>
<td>Graph intersection, union, difference</td>
</tr>
<tr>
<td>To combine multiple protein interaction graphs</td>
<td>Majority graph query</td>
</tr>
<tr>
<td>To construct pathways from individual reactions</td>
<td>Graph composition</td>
</tr>
<tr>
<td>To connect pathways, metabolism of co-existing organisms</td>
<td>Graph composition</td>
</tr>
<tr>
<td>Identify “important” paths from nutrients to chemical outputs</td>
<td>Shortest path queries</td>
</tr>
<tr>
<td>Find all products ultimately derived from a particular reaction</td>
<td>Transitive Closure</td>
</tr>
<tr>
<td>Observe multiple products are co-regulated</td>
<td>Least common ancestor</td>
</tr>
<tr>
<td>To find biopathways graph motifs</td>
<td>Frequent subgraph recognition</td>
</tr>
<tr>
<td>Chemical info retrieval</td>
<td>Subgraph isomorphism</td>
</tr>
<tr>
<td>Kinaze enzyme</td>
<td>Subgraph homomorphism</td>
</tr>
<tr>
<td>Enzyme taxonomies</td>
<td>Subsumption testing</td>
</tr>
<tr>
<td>To find biopathways graph motifs</td>
<td>Frequent subgraph recognition</td>
</tr>
</tbody>
</table>
## Use cases: Web domain

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Graph Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is/are the most cited paper/s?</td>
<td>Degree of a node</td>
</tr>
<tr>
<td>What is the influence of article D?</td>
<td>Paths</td>
</tr>
<tr>
<td>What is the Erdös distance between authors X and author Y?</td>
<td>Distance</td>
</tr>
<tr>
<td>Are suspects A and B related?</td>
<td>Paths</td>
</tr>
<tr>
<td>All relatives of degree one of Alice</td>
<td>Adjacency</td>
</tr>
</tbody>
</table>

### Friend Of A Friend (FOAF)

**Brickley & Miller**

```
foaf:Person  rdf:type foaf:Person

foaf:name  foaf:Person

foaf:name  rdf:type foaf:Person

foaf:name  foaf:Person

mailto:mm@example.com

mailto:dd@example.com
```

Michael Souris

Donald Canard
Example: genealogy data and diagram

<table>
<thead>
<tr>
<th>NAME</th>
<th>LASTNAME</th>
<th>PERSON</th>
<th>PARENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>George</td>
<td>Jones</td>
<td>Julia</td>
<td>George</td>
</tr>
<tr>
<td>Ana</td>
<td>Stone</td>
<td>Julia</td>
<td>Ana</td>
</tr>
<tr>
<td>Julia</td>
<td>Jones</td>
<td>David</td>
<td>James</td>
</tr>
<tr>
<td>James</td>
<td>Deville</td>
<td>David</td>
<td>Julia</td>
</tr>
<tr>
<td>David</td>
<td>Deville</td>
<td>Mary</td>
<td>James</td>
</tr>
<tr>
<td>Mary</td>
<td>Deville</td>
<td>Mary</td>
<td>Julia</td>
</tr>
</tbody>
</table>

Fig. 5. A genealogy diagram (right-hand side) represented as two tables (left-hand side) NAME-LASTNAME and PERSON-PARENT (Children inherit the lastname of the father just for modeling purposes).

Fig. 6. Logical Data Model. The schema (on the left) uses two basic type nodes for representing data values (N and L), and two product type nodes (NL and PP) to establish relations among data values in a relational style. The instance (on the right) is a collection of tables, one for each node of the schema. Note that internal nodes use pointers (names) to make reference to basic and set data values defined by other nodes.

The Hypernode db-model was described in a sequence of papers [Levene and Poulovassilis 1990; Poulovassilis and Levene 1994; Levene and Loizou 1995]. A hypernode is a directed graph whose nodes can themselves be graphs (or hypernodes), allowing nesting of graphs. Hypernodes can be used to represent simple (flat) and complex, objects (hierarchical, composite, and cyclic) as well as mappings and records. A key feature is its inherent ability to encapsulate information.
Fig. 5. A genealogy diagram (right-hand side) represented as two tables (left-hand side) NAME-LASTNAME and PERSON-PARENT (Children inherit the lastname of the father just for modeling purposes).

Fig. 6. Logical Data Model. The schema (on the left) uses two basic type nodes for representing data values (N and L), and two product type nodes (NL and PP) to establish relations among data values in a relational style. The instance (on the right) is a collection of tables, one for each node of the schema. Note that internal nodes use pointers (names) to make reference to basic and set data values defined by other nodes.

A.2. Hypernode Model

The Hypernode db-model was described in a sequence of papers [Levene and Poulovassilis 1990; Poulovassilis and Levene 1994; Levene and Loizou 1995]. A hypernode is a directed graph whose nodes can themselves be graphs (or hypernodes), allowing nesting of graphs. Hypernodes can be used to represent simple (flat) and complex, objects (hierarchical, composite, and cyclic) as well as mappings and records. A key feature is its inherent ability to encapsulate information.

The Hypernode model was introduced by Levene and Poulovassilis [1990], who define the model and a declarative logic-based language structured as a sequence of instructions (hypernode programs), used for querying and updating hypernodes. The implementation of a storage system based on the hypernode model is presented in Tuv et al. [1992].

In a second version Poulovassilis and Levene [1994], introduce the notion of schema and type checking via the idea of types (primitive and complex), that are also represented by nested graphs. (See an example in Figure 7.) The model is completed with entity and referential integrity constraints over a hypernode repository. Moreover it presents a rule-based query language called Hyperlog, which can support both querying and browsing with derivations as well as database updates, and is intractable in the general case.

A third version of the model [Levene and Loizou 1995] discusses a set of constraints (entity, referential, and semantic) over hypernode databases and introduces the concept...
Fig. 5. A genealogy diagram (right-hand side) represented as two tables (left-hand side) NAME-LASTNAME and PERSON-PARENT (Children inherit the lastname of the father just for modeling purposes).

Fig. 6. Logical Data Model. The schema (on the left) uses two basic type nodes for representing data values (N and L), and two product type nodes (NL and PP) to establish relations among data values in a relational style. The instance (on the right) is a collection of tables, one for each node of the schema. Note that internal nodes use pointers (names) to make reference to basic and set data values defined by other nodes.

Level. Due to the fact that the model is a generalization of other models, like the relational model, their techniques or properties can be translated into the generalized model. A relevant example is the definition of integrity constraints.

A.2. Hypernode Model

The Hypernode db-model was described in a sequence of papers [Levene and Poulovassilis 1990; Poulovassilis and Levene 1994; Levene and Loizou 1995]. A hypernode is a directed graph whose nodes can themselves be graphs (or hypernodes), allowing nesting of graphs. Hypernodes can be used to represent simple (flat) and complex, objects (hierarchical, composite, and cyclic) as well as mappings and records. A key feature is its inherent ability to encapsulate information.

The Hypernode model was introduced by Levene and Poulovassilis [1990], who define the model and a declarative logic-based language structured as a sequence of instructions (hypernode programs), used for querying and updating hypernodes. The implementation of a storage system based on the hypernode model is presented in Tuv et al. [1992].

In a second version Poulovassilis and Levene [1994], introduce the notion of schema and type checking via the idea of types (primitive and complex), that are also represented by nested graphs. (See an example in Figure 7.) The model is completed with entity and referential integrity constraints over a hypernode repository. Moreover it presents a rule-based query language called Hyperlog, which can support both querying and browsing with derivations as well as database updates, and is intractable in the general case.

A third version of the model [Levene and Loizou 1995] discusses a set of constraints (entity, referential, and semantic) over hypernode databases and introduces the concept of a hypernode repository. The model is completed with entity and referential integrity constraints over a hypernode repository. Moreover it presents a rule-based query language called Hyperlog, which can support both querying and browsing with derivations as well as database updates, and is intractable in the general case.
A genealogy diagram (right-hand side) represented as two tables (left-hand side) NAME-LASTNAME and PERSON-PARENT (Children inherit the lastname of the father just for modeling purposes).

Fig. 6. Logical Data Model. The schema (on the left) uses two basic type nodes for representing data values (N and L), and two product type nodes (NL and PP) to establish relations among data values in a relational style. The instance (on the right) is a collection of tables, one for each node of the schema. Note that internal nodes use pointers (names) to make reference to basic and set data values defined by other nodes.

Due to the fact that the model is a generalization of other models, like the relational model, their techniques or properties can be translated into the generalized model. A relevant example is the definition of integrity constraints.

A.2. Hypernode Model

The Hypernode db-model was described in a sequence of papers [Levene and Poulovassilis 1990; Poulovassilis and Levene 1994; Levene and Loizou 1995]. A hypernode is a directed graph whose nodes can themselves be graphs (or hypernodes), allowing nesting of graphs. Hypernodes can be used to represent simple (flat) and complex, objects (hierarchical, composite, and cyclic) as well as mappings and records. A key feature is its inherent ability to encapsulate information.

The Hypernode model was introduced by Levene and Poulovassilis [1990], who define the model and a declarative logic-based language structured as a sequence of instructions (hypernode programs), used for querying and updating hypernodes. The implementation of a storage system based on the hypernode model is presented in Tuv et al. [1992].

In a second version Poulovassilis and Levene [1994], introduce the notion of schema and type checking via the idea of types (primitive and complex), that are also represented by nested graphs. (See an example in Figure 7.) The model is completed with entity and referential integrity constraints over a hypernode repository. Moreover it presents a rule-based query language called Hyperlog, which can support both querying and browsing with derivations as well as database updates, and is intractable in the general case.

A third version of the model [Levene and Loizou 1995] discusses a set of constraints (entity, referential, and semantic) over hypernode databases and introduces the concept...
Fig. 5. A genealogy diagram (right-hand side) represented as two tables (left-hand side) NAME-LASTNAME and PERSON-PARENT (Children inherit the lastname of the father just for modeling purposes).

Fig. 6. Logical Data Model. The schema (on the left) uses two basic type nodes for representing data values (N and L), and two product type nodes (NL and PP) to establish relations among data values in a relational style. The instance (on the right) is a collection of tables, one for each node of the schema. Note that internal nodes use pointers (names) to make reference to basic and set data values defined by other nodes.

Due to the fact that the model is a generalization of other models, like the relational model, their techniques or properties can be translated into the generalized model. A relevant example is the definition of integrity constraints.

A.2. Hypernode Model

The Hypernode db-model was described in a sequence of papers [Levene and Poulovassilis 1990; Poulovassilis and Levene 1994; Levene and Loizou 1995]. A hypernode is a directed graph whose nodes can themselves be graphs (or hypernodes), allowing nesting of graphs. Hypernodes can be used to represent simple (flat) and complex, objects (hierarchical, composite, and cyclic) as well as mappings and records. A key feature is its inherent ability to encapsulate information.

The Hypernode model was introduced by Levene and Poulovassilis [1990], who define the model and a declarative logic-based language structured as a sequence of instructions (hypernode programs), used for querying and updating hypernodes. The implementation of a storage system based on the hypernode model is presented in Tuv et al. [1992].

In a second version Poulovassilis and Levene [1994], introduce the notion of schema and type checking via the idea of types (primitive and complex), that are also represented by nested graphs. (See an example in Figure 7.) The model is completed with entity and referential integrity constraints over a hypernode repository. Moreover it presents a rule-based query language called Hyperlog, which can support both querying and browsing with derivations as well as database updates, and is intractable in the general case.

A third version of the model [Levene and Loizou 1995] discusses a set of constraints (entity, referential, and semantic) over hypernode databases and introduces the concept...
Property Graph data model

Fig. 5. A genealogy diagram (right-hand side) represented as two tables (left-hand side) NAME-LASTNAME and PERSON-PARENT (Children inherit the lastname of the father just for modeling purposes).

Fig. 6. Logical Data Model. The schema (on the left) uses two basic type nodes for representing data values (N and L), and two product type nodes (NL and PP) to establish relations among data values in a relational style. The instance (on the right) is a collection of tables, one for each node of the schema. Note that internal nodes use pointers (names) to make reference to basic and set data values defined by other nodes.

A.2. Hypernode Model

The Hypernode db-model was described in a sequence of papers [Levene and Poulovassilis 1990; Poulovassilis and Levene 1994; Levene and Loizou 1995]. A hypernode is a directed graph whose nodes can themselves be graphs (or hypernodes), allowing nesting of graphs. Hypernodes can be used to represent simple (flat) and complex, objects (hierarchical, composite, and cyclic) as well as mappings and records. A key feature is its inherent ability to encapsulate information.

The Hypernode model was introduced by Levene and Poulovassilis [1990], who define the model and a declarative logic-based language structured as a sequence of instructions (hypernode programs), used for querying and updating hypernodes. The implementation of a storage system based on the hypernode model is presented in Tuv et al. [1992].

In a second version Poulovassilis and Levene [1994], introduce the notion of schema and type checking via the idea of types (primitive and complex), that are also represented by nested graphs. (See an example in Figure 7.) The model is completed with entity and referential integrity constraints over a hypernode repository. Moreover it presents a rule-based query language called Hyperlog, which can support both querying and browsing with derivations as well as database updates, and is intractable in the general case.

A third version of the model [Levene and Loizou 1995] discusses a set of constraints (entity, referential, and semantic) over hypernode databases and introduces the concept...
Graph Query Language: Database theoretician’s view

as a graph data model?

as a relational model?
**Theorem (Gaifman).** A property of graphs is expressible by a closed first order formula iff it is equivalent to a combination of properties of the form

$$\exists v_1, \ldots, v_s \left[ \bigwedge_{1 \leq i \leq s} P(N(v_i, r)) \land \bigwedge_{1 \leq i < j \leq s} d(v_i, v_j) > 2r \right]$$

where $v_1, \ldots, v_s$ denote vertices and $d(x,y)$ denotes distance
Theorem (Gaifman). A property of graphs is expressible by a closed first order formula iff it is equivalent to a combination of properties of the form

$$\exists v_1, \ldots, v_s \left[ \bigwedge_{1 \leq i \leq s} P(N(v_i, r)) \land \bigwedge_{1 \leq i < j \leq s} d(v_i, v_j) > 2r \right]$$

where $v_1, \ldots, v_s$ denote vertices and $d(x, y)$ is distance.
### Classical graph query languages

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>Neighborhoods</th>
<th>Adjacent Edges</th>
<th>Degree of a Node</th>
<th>Path</th>
<th>Fixed-length path</th>
<th>Distance</th>
<th>Diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph Query Language</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Classical graph query languages
Some naive questions

1. What is the right balance between expresiveness and complexity for graphs
2. What are the core operations on graphs?
3. What are the “good” families of applications?
4. “Interface” between small-graph and big-graph data management (btw. Where is the frontier?)
5. Graph theory: How that interact with graph data management?
less naive questions

1. Why this obsession with paths?
2. Is the research program of “graph data management” too ambitious?
3. Next step: consider relations between families of nodes.
4. What is the right machinery for graphs? Logic? Data mining / statistics? Programming? … is there a “right” machinery?