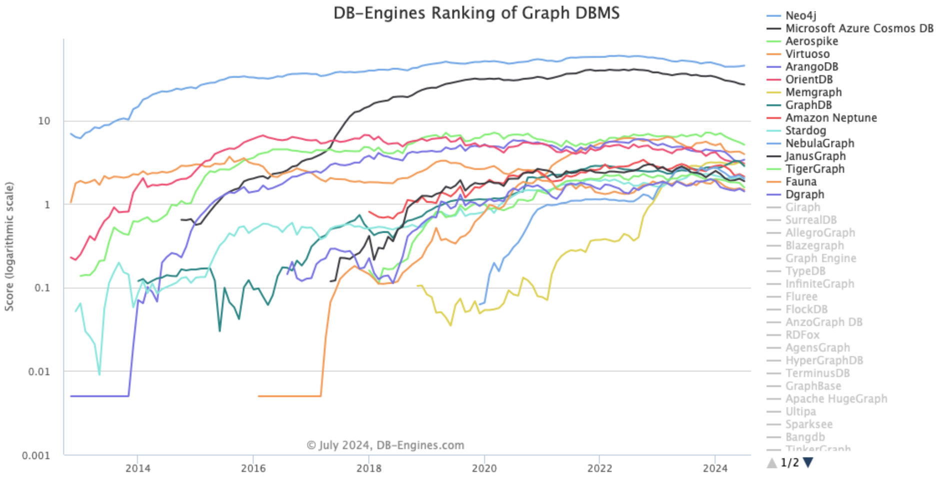
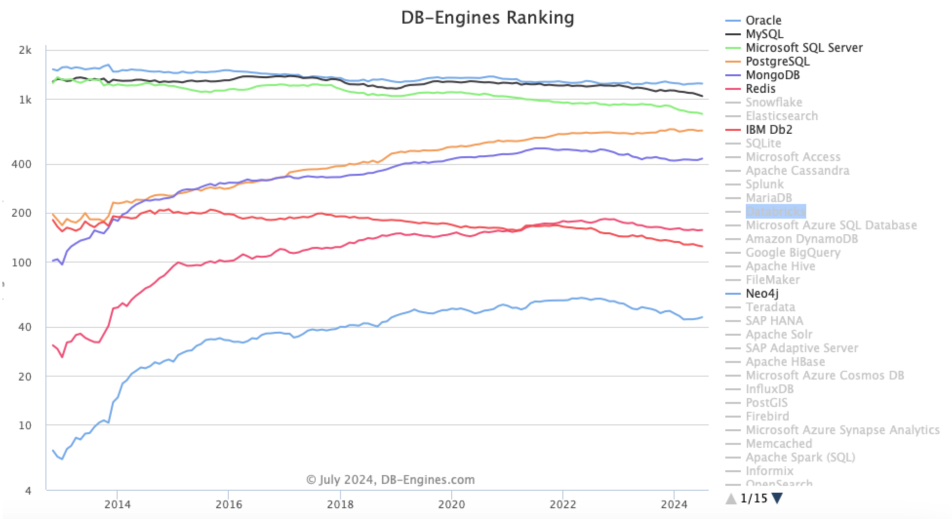

DB-Engines Ranking of Graph DBMS
Score (logarithmic scale)
10
1
0.1
0.01
0.001
2014
2016
2018
2020
2022
2024
© July 2024, DB-Engines.com
Neo4j
Microsoft Azure Cosmos DB
Aerospike
Virtuoso
ArangoDB
OrientDB
Memgraph
GraphDB
Amazon Neptune
Stardog
NebulaGraph
JanusGraph
TigerGraph
Fauna
Dgraph
Giraph
SurrealDB
AllegroGraph
Blazegraph
Graph Engine
TypeDB
InfiniteGraph
Fluree
FlockDB
AnzoGraph DB
RDFox
AgensGraph
HyperGraphDB
TerminusDB
GraphBase
Apache HugeGraph
Ultipa
Sparksee
Bangdb
1/2

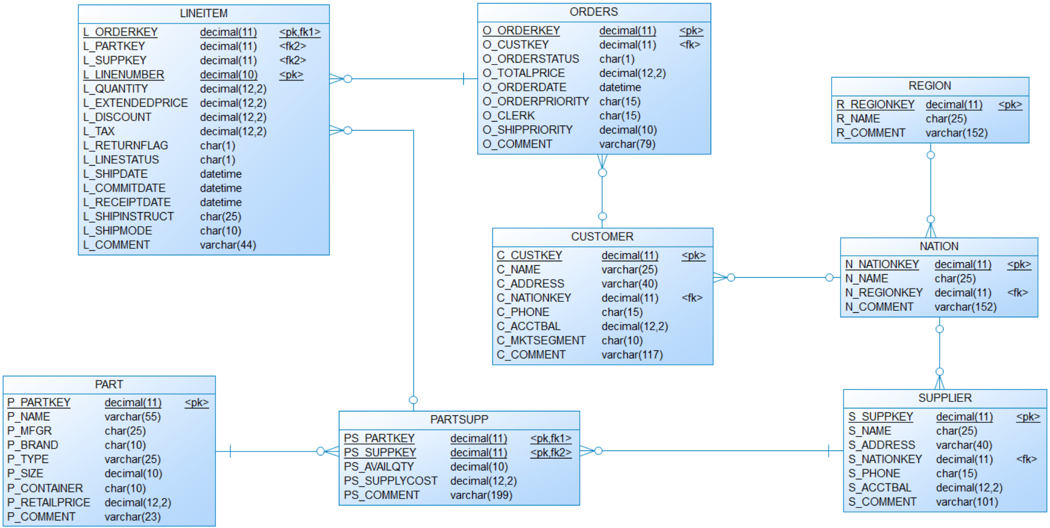
Why are graph databases not so popular yet, apart from historic reasons?



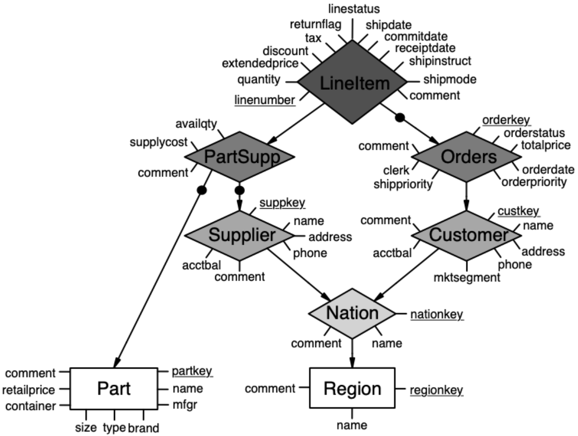
How can we design graph databases well?

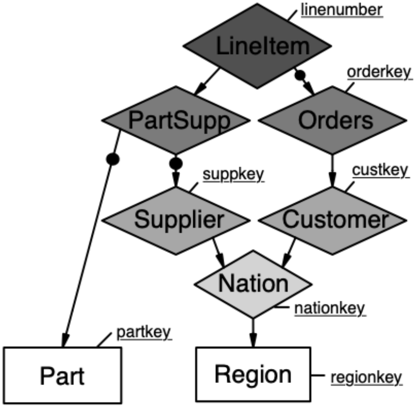
How can we design graph databases well?

The TPC-H Data Model in UML



The TPC-H Data Model as E/R Diagram





Observations

Entity/Relationship diagrams are graphs

Entity/Relationship models capture well-designed databases

Observations

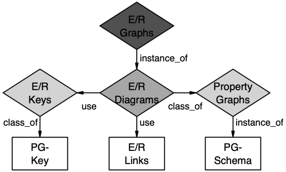
Entity/Relationship diagrams are graphs
Entity/Relationship models capture well-designed databases

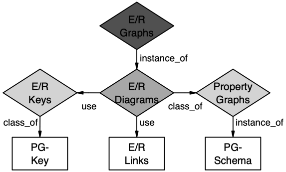
Approach

Explore what E/R modeling has to offer for graph data
Also explore what graph data has to offer for E/R modeling

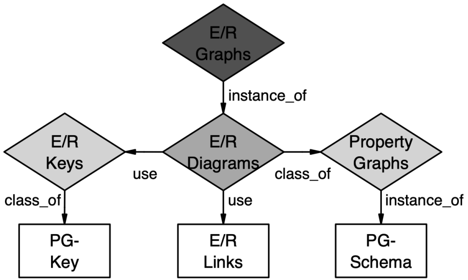
Entity/Relationship Graphs

Unifying Data Modelling and Taking Integrity Management to the Next Level





1. Background
   - E/R Modeling
   - PG-Key and PG-Schema
2. Contributions: "Lean and mean"
   - E/R Diagrams as Property Graphs and Property Graph Models
   - E/R Graphs as Instances of E/R Diagrams
   - Unifying Conceptual, Logical and Graph Modeling
   - Entity and Referential Integrity Management
   - Relational, Mixed, and Graph Semantics of E/R Graphs
3. Experiments
   - Entity Integrity
   - Referential Integrity
4. Summary



Background: Entity/Relationship Modeling

Peter P. Chen: **The Entity-Relationship Model - Toward a Unified View of Data**. *ACM Trans. Database Syst.* 1(1): 9-36 (1976)

Bernhard Thalheim: **Entity-Relationship Modeling - Foundations of Database Technology**. Springer 2000, ISBN 978-3-540-65470-4, pp. I-XII, 1-627

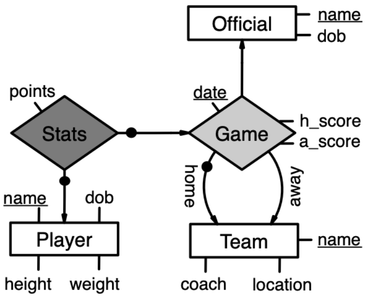
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

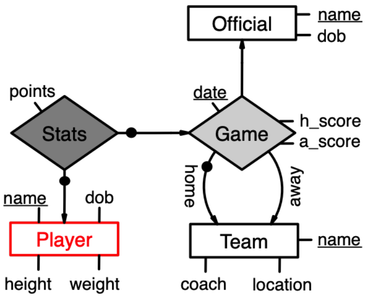
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

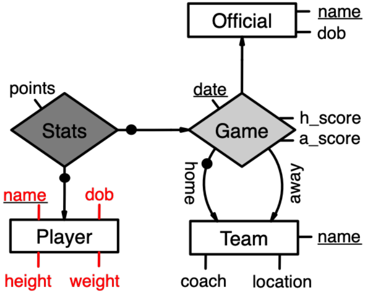
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}{:}\textsc{Team}, \text{away}{:}\textsc{Team}, \textsc{Official}\},$ $\{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}{:}\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

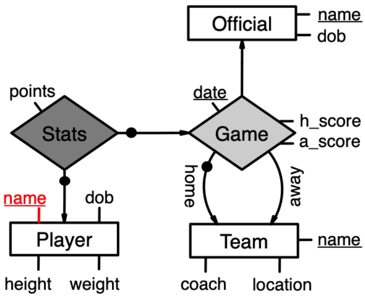
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\},$ $\{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

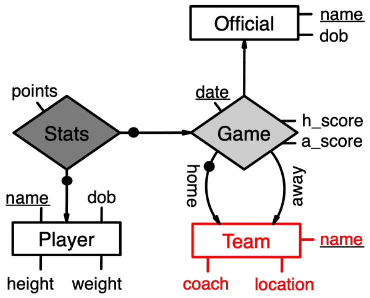
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

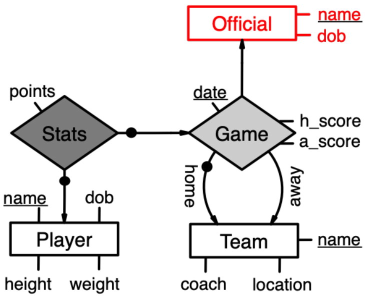
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

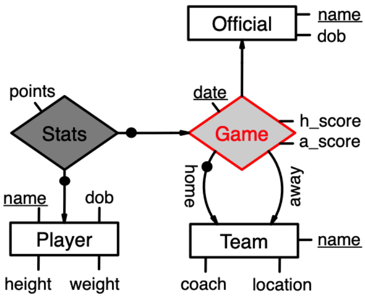
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\text{PLAYER} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\text{TEAM} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\text{OFFICIAL} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\text{GAME} = (\{\text{home:TEAM}, \text{away:TEAM}, \text{OFFICIAL}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home:TEAM}, \text{date}\})$
- $\text{STATS} = (\{\text{GAME}, \text{PLAYER}\}, \{\text{points}\}, \{\text{GAME}, \text{PLAYER}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

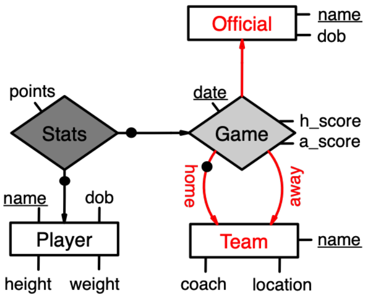
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

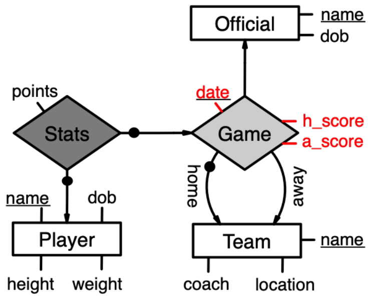
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

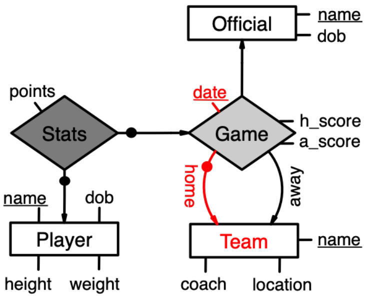
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

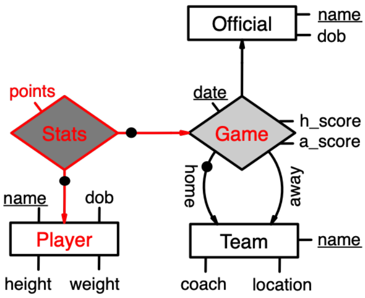
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}{:}\textsc{Team}, \text{away}{:}\textsc{Team}, \textsc{Official}\},$ $\{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}{:}\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

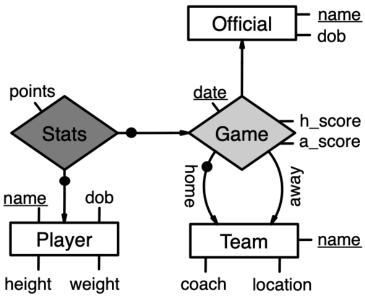
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\},$ $\{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

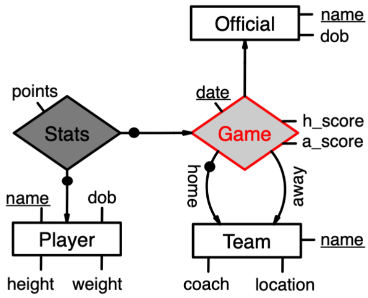
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\},$ $\{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

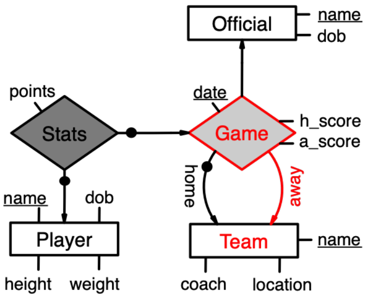
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}{:}\textsc{Team}, \text{away}{:}\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}{:}\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

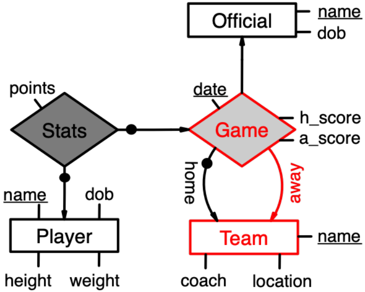
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}{:}\textsc{Team}, \text{away}{:}\textsc{Team}, \textsc{Official}\},$ $\{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}{:}\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

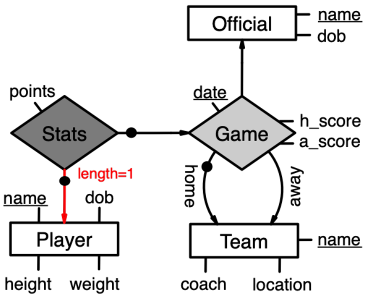
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

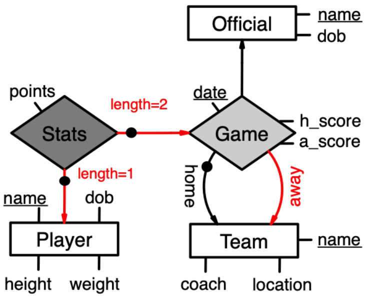
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home:}\textsc{Team}, \text{away:}\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home:}\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

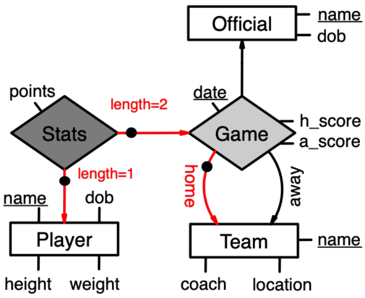
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home:}\textsc{Team}, \text{away:}\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home:}\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

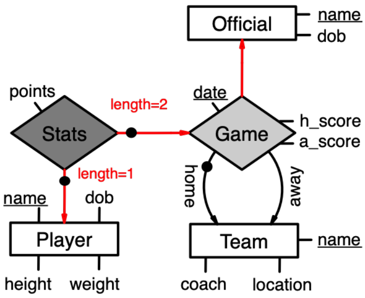
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}:\textsc{Team}, \text{away}:\textsc{Team}, \textsc{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}:\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

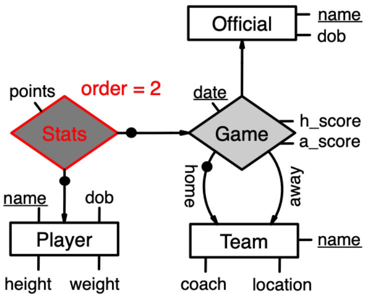
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\textsc{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\textsc{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\textsc{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\textsc{Game} = (\{\text{home}{:}\textsc{Team}, \text{away}{:}\textsc{Team}, \textsc{Official}\},$ $\{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home}{:}\textsc{Team}, \text{date}\})$
- $\textsc{Stats} = (\{\textsc{Game}, \textsc{Player}\}, \{\text{points}\}, \{\textsc{Game}, \textsc{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

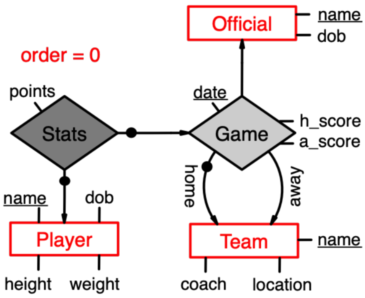
E/R Schemata and their Diagrams

Entity Types $E = (attr(E), id(E))$

- $\text{Player} = (\{\text{name}, \text{dob}, \text{height}, \text{weight}\}, \{\text{name}\})$
- $\text{Team} = (\{\text{name}, \text{coach}, \text{location}\}, \{\text{name}\})$
- $\text{Official} = (\{\text{name}, \text{license}\}, \{\text{name}\})$

Relationship Types $R = (comp(R), attr(R), id(R))$

- $\text{Game} = (\{\text{home:Team}, \text{away:Team}, \text{Official}\}, \{\text{date}, \text{home\_score}, \text{away\_score}\}, \{\text{home:Team}, \text{date}\})$
- $\text{Stats} = (\{\text{Game}, \text{Player}\}, \{\text{points}\}, \{\text{Game}, \text{Player}\})$



E/R Schema and order k of its types

- Set $\mathcal{S}$ of object types (entity and relationship types) whose diagram forms a DAG such that for every $O \in \mathcal{S}$ and every $O' \in comp(O)$, $O' \in \mathcal{S}$
- Length of longest path from a given object type to a leaf is its order k
- Entity types have order $k = 0$

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

E/R (Foreign Keys) Semantics

Player

name	*dob*	*height*	*weight*
Jordan	17/02/1963	1.98m	98kg
Malone	24/07/1963	2.06m	117kg

Team

name	*location*	*coach*
Bulls	United Center	Jackson
Jazz	Delta Center	Sloan

Official

name	*dob*
Crawford	30/08/1951
Evans	08/11/1940

Game

home:Team	away:TEAM	**date**	*h_score*	*a_score*
Jazz	Bulls	12/06/1997	88	90
Bulls	Jazz	14/06/1997	90	86

Stats

Player	**home:Team**	**date**	points
Jordan	Jazz	12/06/1997	38
Malone	Jazz	12/06/1997	19
Jordan	Bulls	14/06/1997	39
Malone	Bulls	14/06/1997	21

E/R instances

- each object type is assigned an object set that satisfies the keys specified
- projections of higher-order objects to their components identify a unique lower-order object from object set of component

Background: PG-Key

Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Keith W. Hare, Jan Hidders, Victor E. Lee, Bei Li, Leonid Libkin, Wim Martens, Filip Murlak, Josh Perryman, Ognjen Savkovic, Michael Schmidt, Juan F. Sequeda, Slawek Staworko, Dominik Tomaszuk: **PG-Keys: Keys for Property Graphs**. *SIGMOD Conference 2021*: 2423-2436

PG-Key: Basic goals

- Flexible and powerful framework for defining key constraints
- Designed by the Linked Data Benchmark Council's Property Graph Schema Working Group
  - with members from industry, academia, and the ISO Graph Query Language standards group
- PG-Key combines basic restrictions to identify, reference and constrain objects, including nodes, edges, and properties

PG-Key: Basic goals

- Flexible and powerful framework for defining key constraints
- Designed by the Linked Data Benchmark Council's Property Graph Schema Working Group
  - with members from industry, academia, and the ISO Graph Query Language standards group
- PG-Key combines basic restrictions to identify, reference and constrain objects, including nodes, edges, and properties

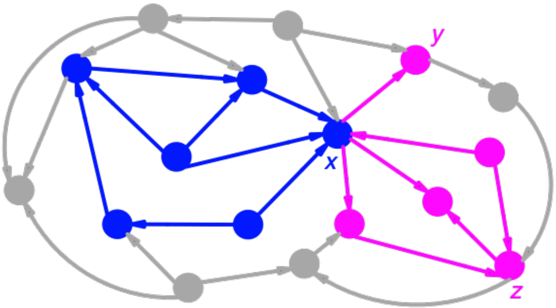
"PG-Keys aims to guide the evolution of the standardization efforts towards making systems more useful, powerful, and expressive".

PG-Key: Informal definition

- Declared on property graph

PG-Key: Informal definition

- Declared on property graph
- Key scope
- Descriptor of values

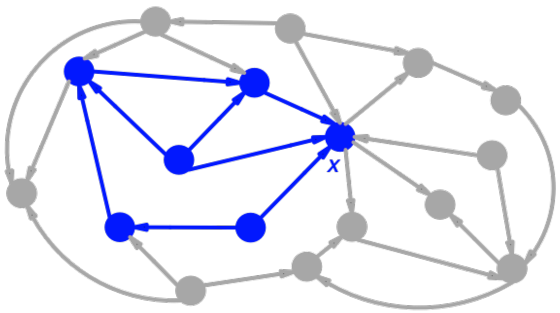


PG-Key: Informal definition

- Declared on property graph
- Key scope
- Descriptor of values

Key scope

FOR x WITHIN



PG-Key: Informal definition

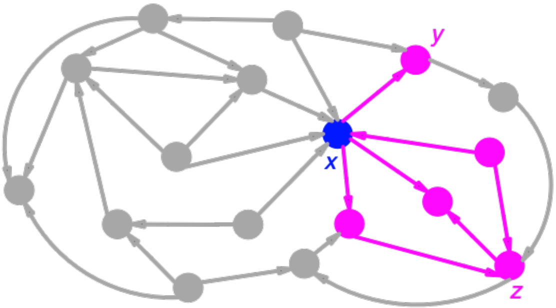
- Declared on property graph
- Key scope
- Descriptor of values

Key scope

FOR x WITHIN

Descriptor of key values

IDENTIFIER y, z WITHIN



PG-Key: Informal definition

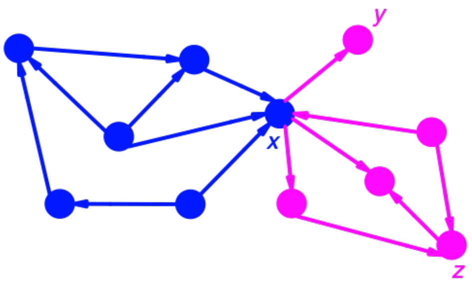
- Declared on property graph
- Key scope
- Descriptor of values

Key scope

FOR x WITHIN

Descriptor of key values

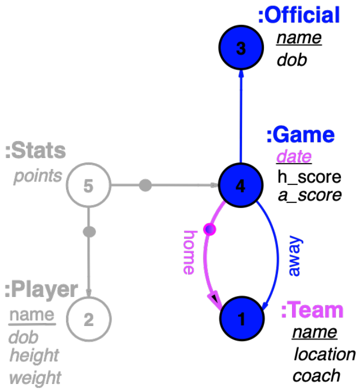
IDENTIFIER y, z WITHIN



PG-Key: Example

FOR g WITHIN
  (g:Game)-[home]->(t:Team),
  (g:Game)-[away]->(:Team),
  (g:Game)-[ ]->(:Official)
IDENTIFIER g**.date**, t

Each game between a home and away team refereed by some official **is identified by the date and home team**



PG-Key: Identify, reference, and constrain objects

Objects comprise nodes, relationships, or properties

PG-Key: Identify, reference, and constrain objects

Objects comprise nodes, relationships, or properties

IDENTIFIER means

- EXCLUSIVE: no objects in the scope share a key value
- MANDATORY: each object in the scope has at least one key value
- SINGLETON: each object in the scope has at most one key value

PG-Key: Identify, reference, and constrain objects

Objects comprise nodes, relationships, or properties

IDENTIFIER means

- EXCLUSIVE: no objects in the scope share a key value
- MANDATORY: each object in the scope has at least one key value
- SINGLETON: each object in the scope has at most one key value

- EXCLUSIVE means UNIQUE in SQL
- MANDATORY means NOT NULL in SQL
- SINGLETON ensured by first normal form

Background: PG-Schema

Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, Dusan Zivkovic: **PG-Schema: Schemas for Property Graphs**. *Proc. ACM Manag. Data* 1(2): 198:1-198:25 (2023)

PG-Schema: Basic goals

- Community-based effort of academics and practitioners to help with standardization efforts for a graph query language, in particular schema support
- PG-Schema aims at providing a simple yet powerful formalism for specifying property graph schemata, including flexible type definitions supporting multi-inheritance, and expressive constraints
- The formal syntax and semantics of PG-Schema is aimed at meeting principled design requirements of contemporary property graph management

PG-Schema: Basic goals

- Community-based effort of academics and practitioners to help with standardization efforts for a graph query language, in particular schema support
- PG-Schema aims at providing a simple yet powerful formalism for specifying property graph schemata, including flexible type definitions supporting multi-inheritance, and expressive constraints
- The formal syntax and semantics of PG-Schema is aimed at meeting principled design requirements of contemporary property graph management

PG-Schema "deliberately target minimal data modeling capabilities and as a reference point ... take the most basic variant of Entity-Relationship (ER) diagrams ... as the ultimate lower bound in the expressiveness of conceptual modelling languages".

PG-Schema: Main features

Domain, Entity, Referential Integrity & Beyond

- Node types, Edge types, Constraints

PG-Schema: Main features

Domain, Entity, Referential Integrity & Beyond

- Node types, Edge types, Constraints

Design features

- Simplicity (easy to understand, validate, & generate; enables partial validation):
  - Node types $\rightarrow$ Edge types $\rightarrow$ Constraints

PG-Schema: Main features

Domain, Entity, Referential Integrity & Beyond

- Node types, Edge types, Constraints

Design features

- Simplicity (easy to understand, validate, & generate; enables partial validation):
  - Node types $\rightarrow$ Edge types $\rightarrow$ Constraints
- Compositionality (reusability, conciseness, modeling power):
  - Union, intersection and abstract types for inheritance and more

PG-Schema: Main features

Domain, Entity, Referential Integrity & Beyond

- Node types, Edge types, Constraints

Design features

- Simplicity (easy to understand, validate, & generate; enables partial validation):
  - Node types $\rightarrow$ Edge types $\rightarrow$ Constraints
- Compositionality (reusability, conciseness, modeling power):
  - Union, intersection and abstract types for inheritance and more
- Versatility (schema first, partial schema, flexible schema):
  - in strict schemata, elements must type and constraints must hold
  - in loose schemata, elements may not type but constraints hold for typed elements

PG-Schema: Main features

Domain, Entity, Referential Integrity & Beyond

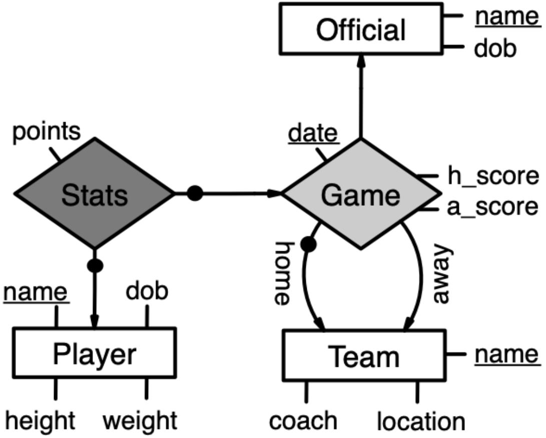
- Node types, Edge types, Constraints

Design features

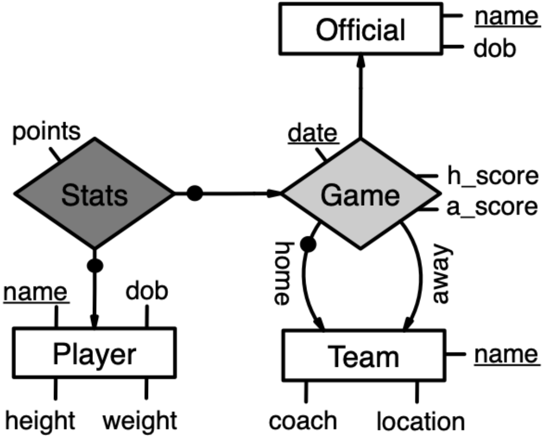
- Simplicity (easy to understand, validate, & generate; enables partial validation):
  - Node types → Edge types → Constraints
- Compositionality (reusability, conciseness, modeling power):
  - Union, intersection and abstract types for inheritance and more
- Versatility (schema first, partial schema, flexible schema):
  - in strict schemata, elements must type and constraints must hold
  - in loose schemata, elements may not type but constraints hold for typed elements
  - closed types allow only explicitly mentioned or inherited labels and properties
  - open types allow arbitrary additional labels and properties

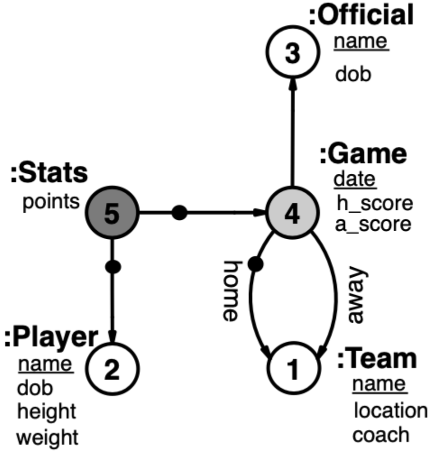
E/R Diagrams are Property Graphs and PG Models

Visual: E/R Diagrams as Property Graphs



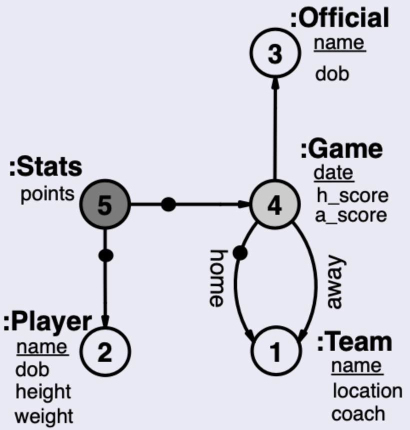
Visual: E/R Diagrams as Property Graphs





Formal definition: E/R Diagrams as Property Graphs

E/R Diagram D

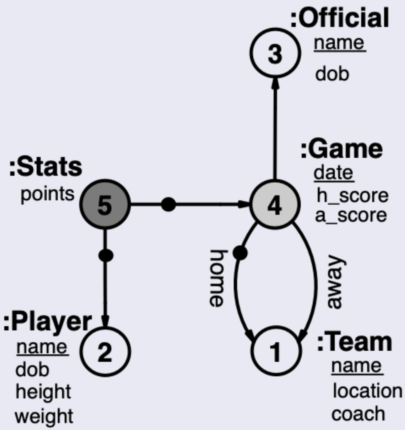


Formal Property Graph $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ for D

- $V_D := \{i_1, i_2, i_3, i_4, i_5\}$,

Formal definition: E/R Diagrams as Property Graphs

E/R Diagram D

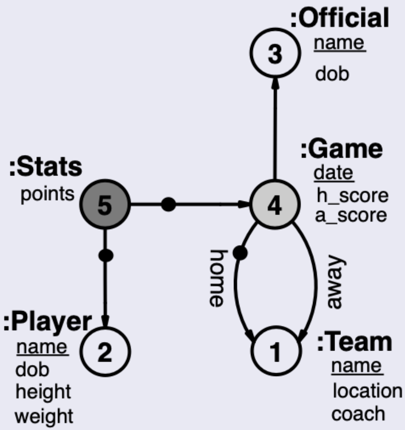


Formal Property Graph $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ for D

- $V_D := \{i_1, i_2, i_3, i_4, i_5\}$,
- $E_D := \{i_{(4,1,\text{home})}, i_{(4,1,\text{away})}, i_{(4,3)}, i_{(5,2)}, i_{(5,4)}\}$,

Formal definition: E/R Diagrams as Property Graphs

E/R Diagram D



Formal Property Graph $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ for D

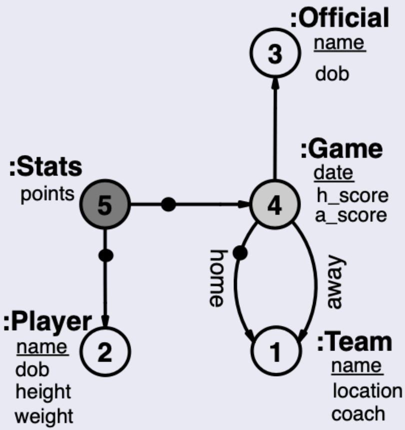
- $V_D := \{i_1, i_2, i_3, i_4, i_5\}$,
- $E_D := \{i_{(4,1,\text{home})}, i_{(4,1,\text{away})}, i_{(4,3)}, i_{(5,2)}, i_{(5,4)}\}$,
- $\eta_D : E_D \to V_D \times V_D$:

  $i_{(4,1,\text{home})} \mapsto (i_4, i_1)$, $i_{(4,1,\text{away})} \mapsto (i_4, i_1)$,

  $i_{(4,3)} \mapsto (i_4, i_3)$, $i_{(5,2)} \mapsto (i_5, i_2)$, $i_{(5,4)} \mapsto (i_5, i_4)$

Formal definition: E/R Diagrams as Property Graphs

E/R Diagram D

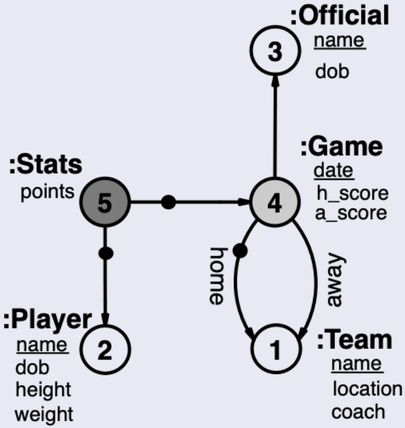


Formal Property Graph $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ for D

- $V_D := \{i_1, i_2, i_3, i_4, i_5\}$,
- $E_D := \{i_{(4,1,\text{home})}, i_{(4,1,\text{away})}, i_{(4,3)}, i_{(5,2)}, i_{(5,4)}\}$,
- $\eta_D : E_D \to V_D \times V_D$:
  $i_{(4,1,\text{home})} \mapsto (i_4, i_1)$, $i_{(4,1,\text{away})} \mapsto (i_4, i_1)$,
  $i_{(4,3)} \mapsto (i_4, i_3)$, $i_{(5,2)} \mapsto (i_5, i_2)$, $i_{(5,4)} \mapsto (i_5, i_4)$
- $\lambda_D : V_D \cup E_D \to \mathcal{P}(\mathcal{L})$:
  $i_1 \mapsto \{\textsc{Team}\}$, $i_2 \mapsto \{\textsc{Player}\}$, $i_3 \mapsto \{\textsc{Official}\}$,
  $i_4 \mapsto \{\textsc{Game}\}$, $i_5 \mapsto \{\textsc{Stats}\}$, $i_{(4,1,\text{home})} \mapsto \{\bullet, \text{home}\}$,
  $i_{(4,1,\text{away})} \mapsto \{\text{away}\}$, $i_{(5,2)} \mapsto \{\bullet\}$, $i_{(5,4)} \mapsto \{\bullet\}$, and

Formal definition: E/R Diagrams as Property Graphs

E/R Diagram D



Formal Property Graph $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ for D

- $V_D := \{i_1, i_2, i_3, i_4, i_5\}$,
- $E_D := \{i_{(4,1,\text{home})}, i_{(4,1,\text{away})}, i_{(4,3)}, i_{(5,2)}, i_{(5,4)}\}$,
- $\eta_D : E_D \to V_D \times V_D$:
  $i_{(4,1,\text{home})} \mapsto (i_4, i_1)$, $i_{(4,1,\text{away})} \mapsto (i_4, i_1)$,
  $i_{(4,3)} \mapsto (i_4, i_3)$, $i_{(5,2)} \mapsto (i_5, i_2)$, $i_{(5,4)} \mapsto (i_5, i_4)$
- $\lambda_D : V_D \cup E_D \to \mathcal{P}(\mathcal{L})$:
  $i_1 \mapsto \{\text{TEAM}\}$, $i_2 \mapsto \{\text{PLAYER}\}$, $i_3 \mapsto \{\text{OFFICIAL}\}$,
  $i_4 \mapsto \{\text{GAME}\}$, $i_5 \mapsto \{\text{STATS}\}$, $i_{(4,1,\text{home})} \mapsto \{\bullet, \text{home}\}$,
  $i_{(4,1,\text{away})} \mapsto \{\text{away}\}$, $i_{(5,2)} \mapsto \{\bullet\}$, $i_{(5,4)} \mapsto \{\bullet\}$, and
- $\nu_D : (V_D \cup E_D) \times \mathcal{K}_D \to \mathcal{N}_D$:
  $(i_1, \text{name}) \mapsto \texttt{String}$, $(i_1, \text{location}) \mapsto \texttt{String}_\perp$, $(i_1, \text{coach}) \mapsto \texttt{String}_\perp$,
  $(i_2, \text{name}) \mapsto \texttt{String}$, $(i_2, \text{dob}) \mapsto \texttt{Date}_\perp$, $(i_2, \text{height}) \mapsto \texttt{String}_\perp$,
  $(i_2, \text{weight}) \mapsto \texttt{String}_\perp$,
  $(i_3, \text{name}) \mapsto \texttt{String}$, $(i_3, \text{dob}) \mapsto \texttt{Date}_\perp$,
  $(i_4, \text{date}) \mapsto \texttt{Date}$, $(i_4, \text{h\_score}) \mapsto \texttt{Int}_\perp$, $(i_4, \text{a\_score}) \mapsto \texttt{Int}_\perp$,
  $(i_5, \text{points}) \mapsto \texttt{Int}$.

Formal Translation: Base Sets

E/R Diagram $D = (V, E)$ for E/R Schema $\mathcal{S}$

- $\mathcal{O}_D = O_V \cup O_E$: set of object identifiers for D composed of the sets
  - O_V with unique identifiers i_o for each vertex in V,
  - O_E with unique identifiers $i_{(o,o',l)}$ for each edge in E,

Formal Translation: Base Sets

E/R Diagram $D = (V, E)$ for E/R Schema $\mathcal{S}$

- $\mathcal{O}_D = O_V \cup O_E$: set of object identifiers for D composed of the sets
  - O_V with unique identifiers i_o for each vertex in V,
  - O_E with unique identifiers $i_{(o,o',l)}$ for each edge in E,
- $\mathcal{L}_D$: set of labels for D comprising
  - names O of object types in V,
  - edge labels in D (resulting from labels l in $l : O' \in comp(O)$), and
  - a distinguished label $\bullet$ for dots on edges (representing key components $O' \in id(O)$).

Formal Translation: Base Sets

E/R Diagram $D = (V, E)$ for E/R Schema $\mathcal{S}$

- $\mathcal{O}_D = O_V \cup O_E$: set of object identifiers for D composed of the sets
  - O_V with unique identifiers i_o for each vertex in V,
  - O_E with unique identifiers $i_{(o,o',l)}$ for each edge in E,
- $\mathcal{L}_D$: set of labels for D comprising
  - names O of object types in V,
  - edge labels in D (resulting from labels l in $l : O' \in comp(O)$), and
  - a distinguished label $\bullet$ for dots on edges (representing key components $O' \in id(O)$).
- $\mathcal{K}_D$: set of properties for D containing the attribute names for all object types in $\mathcal{S}$

Formal Translation: Base Sets

E/R Diagram $D = (V, E)$ for E/R Schema $\mathcal{S}$

- $\mathcal{O}_D = O_V \cup O_E$: set of object identifiers for D composed of the sets
  - O_V with unique identifiers i_o for each vertex in V,
  - O_E with unique identifiers $i_{(o,o',l)}$ for each edge in E,
- $\mathcal{L}_D$: set of labels for D comprising
  - names O of object types in V,
  - edge labels in D (resulting from labels l in $l : O' \in comp(O)$), and
  - a distinguished label $\bullet$ for dots on edges (representing key components $O' \in id(O)$).
- $\mathcal{K}_D$: set of properties for D containing the attribute names for all object types in $\mathcal{S}$
- $\mathcal{N}_D$: the set of values for D with all domains $dom(A)$ for any attribute A of any object type in $\mathcal{S}$, but we distinguish between
  - $dom(A)$ which does not permit null marker occurrences
  - $dom_\perp(A)$ which contains the distinguished null marker symbol $\perp$

Formal Translation: Base Sets

E/R Diagram $D = (V, E)$ for E/R Schema $\mathcal{S}$

- $\mathcal{O}_D = O_V \cup O_E$: set of object identifiers for D composed of the sets
  - O_V with unique identifiers i_o for each vertex in V,
  - O_E with unique identifiers $i_{(o,o',l)}$ for each edge in E,
- $\mathcal{L}_D$: set of labels for D comprising
  - names O of object types in V,
  - edge labels in D (resulting from labels l in $l : O' \in comp(O)$), and
  - a distinguished label $\bullet$ for dots on edges (representing key components $O' \in id(O)$).
- $\mathcal{K}_D$: set of properties for D containing the attribute names for all object types in $\mathcal{S}$
- $\mathcal{N}_D$: the set of values for D with all domains $dom(A)$ for any attribute A of any object type in $\mathcal{S}$, but we distinguish between
  - $dom(A)$ which does not permit null marker occurrences
  - $dom_\perp(A)$ which contains the distinguished null marker symbol $\perp$

Principle of entity integrity: as for primary keys, values for key properties are mandatory (`NOT NULL`)

Formal Translation: Definition of $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$

Given $\mathcal{O}_D$, $\mathcal{L}_D$, $\mathcal{K}_D$, and $\mathcal{N}_D$, the *E/R graph model* $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ of an E/R diagram $D = (V, E)$ is defined as follows:

Formal Translation: Definition of $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$

Given $\mathcal{O}_D$, $\mathcal{L}_D$, $\mathcal{K}_D$, and $\mathcal{N}_D$, the *E/R graph model* $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ of an E/R diagram $D = (V, E)$ is defined as follows:

1. $V_D := O_V$,
2. $E_D := O_E$,

Formal Translation: Definition of $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$

Given $\mathcal{O}_D$, $\mathcal{L}_D$, $\mathcal{K}_D$, and $\mathcal{N}_D$, the *E/R graph model* $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ of an E/R diagram $D = (V, E)$ is defined as follows:

1. $V_D := O_V$,
2. $E_D := O_E$,
3. $\eta_D : E_D \rightarrow V_D \times V_D$ is defined by $\eta_D(i_{(o,o',l)}) = (i_o, i_{o'})$ where i_o identifies node $O \in V$ and $i_{o'}$ identifies node $O' \in V$ for every $((O, O'), l) \in E$,

Formal Translation: Definition of $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$

Given $\mathcal{O}_D$, $\mathcal{L}_D$, $\mathcal{K}_D$, and $\mathcal{N}_D$, the *E/R graph model* $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ of an E/R diagram $D = (V, E)$ is defined as follows:

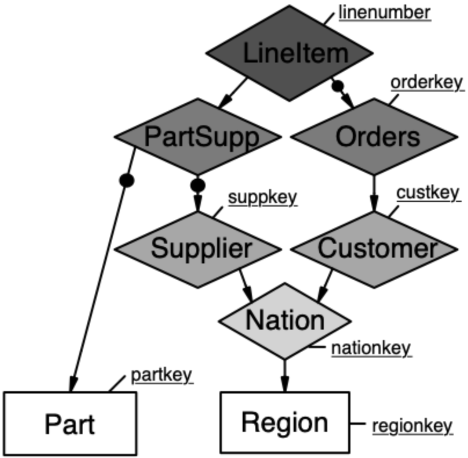
1. $V_D := O_V$,
2. $E_D := O_E$,
3. $\eta_D : E_D \to V_D \times V_D$ is defined by $\eta_D(i_{(o,o',l)}) = (i_o, i_{o'})$ where i_o identifies node $O \in V$ and $i_{o'}$ identifies node $O' \in V$ for every $((O, O'), l) \in E$,
4. $\lambda_D : V_D \cup E_D \to \mathcal{P}(\mathcal{L})$ is defined by $\lambda_D(i_o) = \{O\}$ for $O \in V$ with unique identifier $i_o \in O_V$, $l \in \lambda_D(i_{(o,o',l)})$ for $((O, O'), l) \in E$ with edge label l and unique identifier $i_{(o,o',l)} \in O_E$, and $\bullet \in \lambda_D(i_{(o,o',l)})$ for $((O, O'), l) \in E$ with key component $l : O' \in comp(O) \cap id(O)$ and unique identifier $i_{(o,o',l)} \in O_E$, and

Formal Translation: Definition of $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$

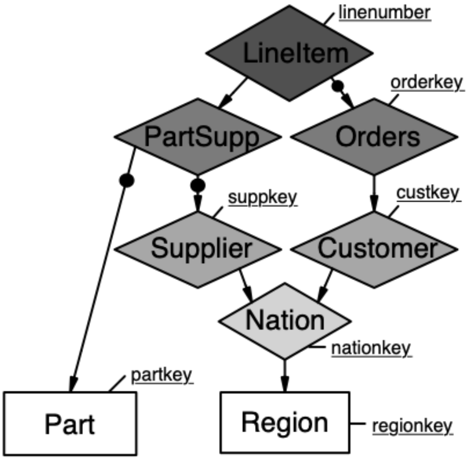
Given $\mathcal{O}_D$, $\mathcal{L}_D$, $\mathcal{K}_D$, and $\mathcal{N}_D$, the *E/R graph model* $\mathcal{G}_D = (V_D, E_D, \eta_D, \lambda_D, \nu_D)$ of an E/R diagram $D = (V, E)$ is defined as follows:

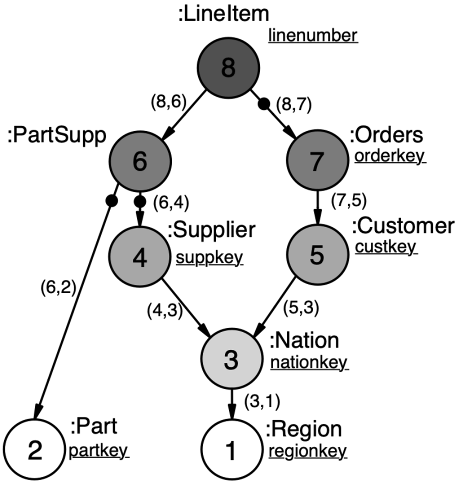
1. $V_D := O_V$,
2. $E_D := O_E$,
3. $\eta_D : E_D \to V_D \times V_D$ is defined by $\eta_D(i_{(o,o',l)}) = (i_o, i_{o'})$ where i_o identifies node $O \in V$ and $i_{o'}$ identifies node $O' \in V$ for every $((O, O'), l) \in E$,
4. $\lambda_D : V_D \cup E_D \to \mathcal{P}(\mathcal{L})$ is defined by $\lambda_D(i_o) = \{O\}$ for $O \in V$ with unique identifier $i_o \in O_V$, $l \in \lambda_D(i_{(o,o',l)})$ for $((O, O'), l) \in E$ with edge label l and unique identifier $i_{(o,o',l)} \in O_E$, and $\bullet \in \lambda_D(i_{(o,o',l)})$ for $((O, O'), l) \in E$ with key component $l : O' \in comp(O) \cap id(O)$ and unique identifier $i_{(o,o',l)} \in O_E$, and
5. $\nu_D : (V_D \cup E_D) \times \mathcal{K}_D \to \mathcal{N}_D$ is defined by $\nu(i_o, A) = dom_\perp(A)$ for all attributes $A \in attr(O) - id(O)$ and all $O \in V$, and $\nu(i_o, A) = dom(A)$ for all key attributes $A \in attr(O) \cap id(O)$ and all $O \in V$.

The TPC-H Data Model as Property Graph

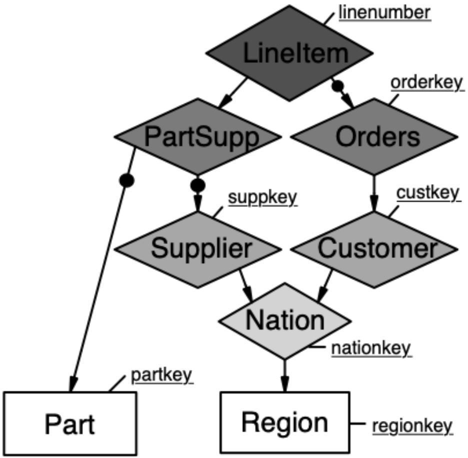


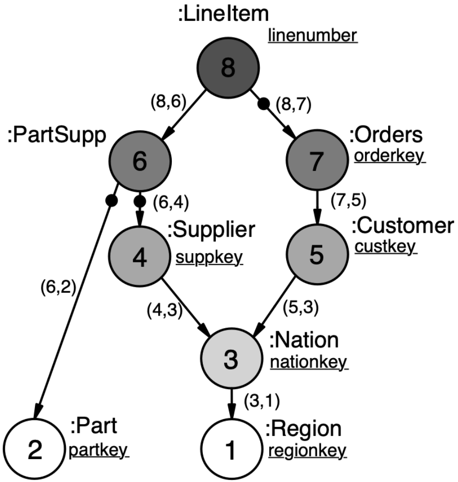
The TPC-H Data Model as Property Graph





The TPC-H Data Model as Property Graph





E/R diagrams are (hierarchical) property graphs that are handy property graph models

Pitstop 1

Contributions

- E/R Schema $\equiv$ E/R Diagrams $\equiv$ E/R Graph Models
- E/R Diagrams are (non-attributed, acyclic) Property Graphs
- Compare:
  - Every XML Schema is an XML document
  - Every E/R Graph Model is a Property graph

Pitstop 1

Contributions

- E/R Schema $\equiv$ E/R Diagrams $\equiv$ E/R Graph Models
- E/R Diagrams are (non-attributed, acyclic) Property Graphs
- Compare:
  - Every XML Schema is an XML document
  - Every E/R Graph Model is a Property graph

E/R Modeling is available for Property Graph Modeling

Pitstop 1

Contributions

- E/R Schema $\equiv$ E/R Diagrams $\equiv$ E/R Graph Models
- E/R Diagrams are (non-attributed, acyclic) Property Graphs
- Compare:
  - Every XML Schema is an XML document
  - Every E/R Graph Model is a Property graph

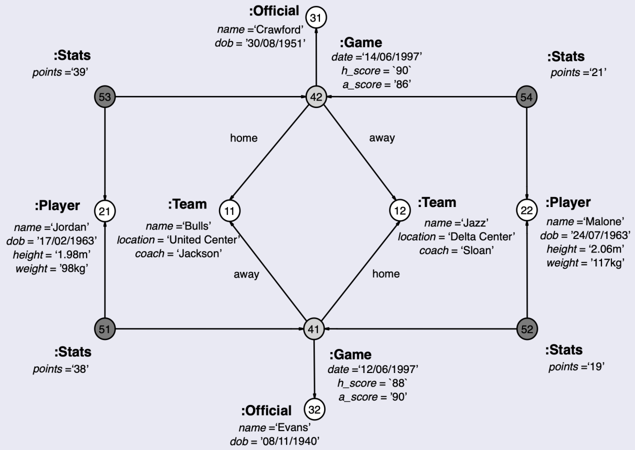
E/R Modeling is available for Property Graph Modeling

Next step: We want a graph semantics for PG modeling with E/R graph model

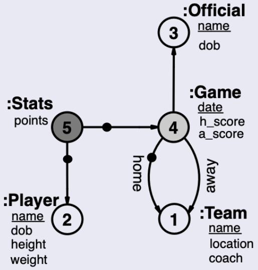
E/R Graphs as Instances of E/R Graph Models

Graph Semantics for E/R Models

E/R Graph

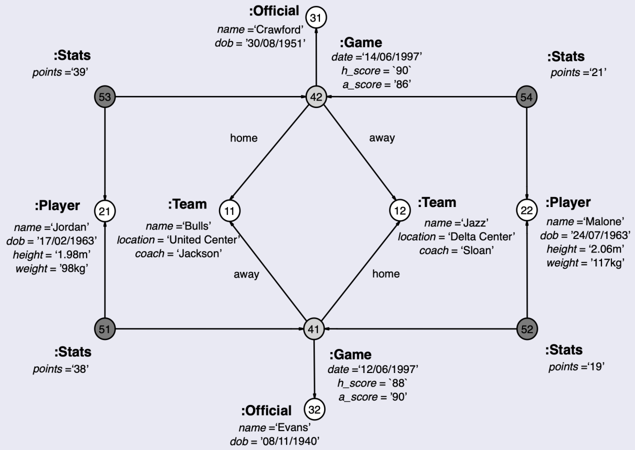


E/R Graph model

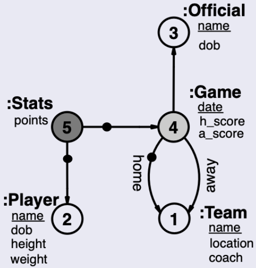


Graph Semantics for E/R Models

E/R Graph

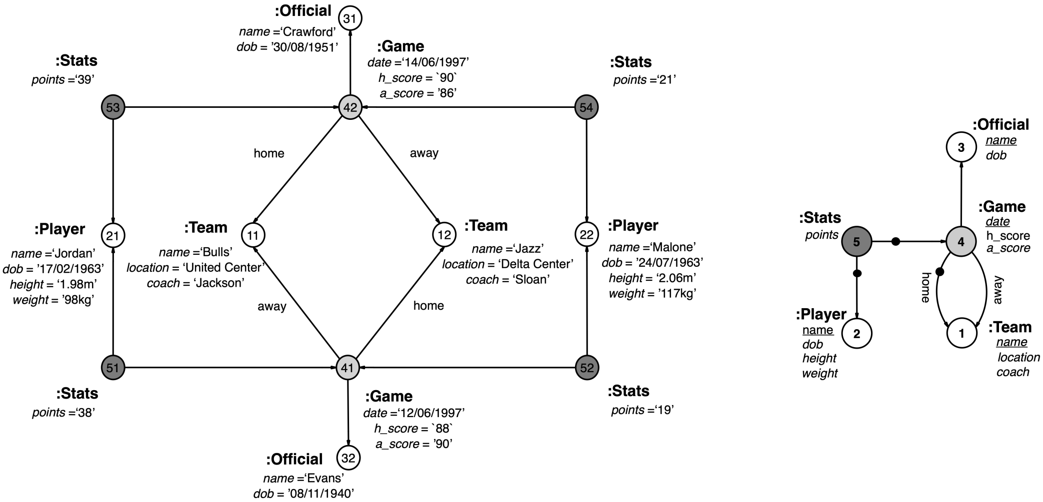


E/R Graph model

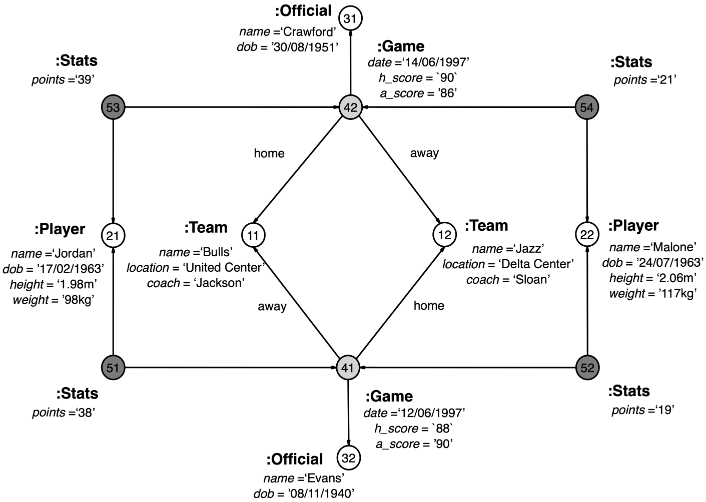


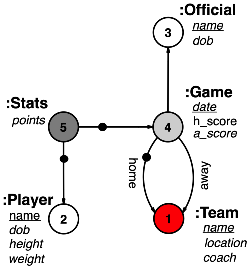
E/R graphs
are instances of
E/R graph models

Illustrating how E/R graphs comply with E/R model graphs

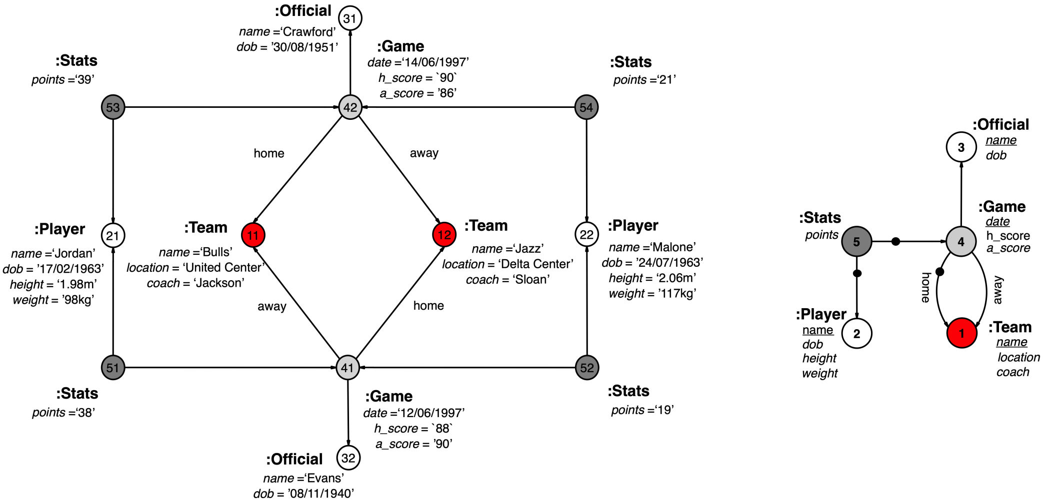


Homomorphism: Example of node image

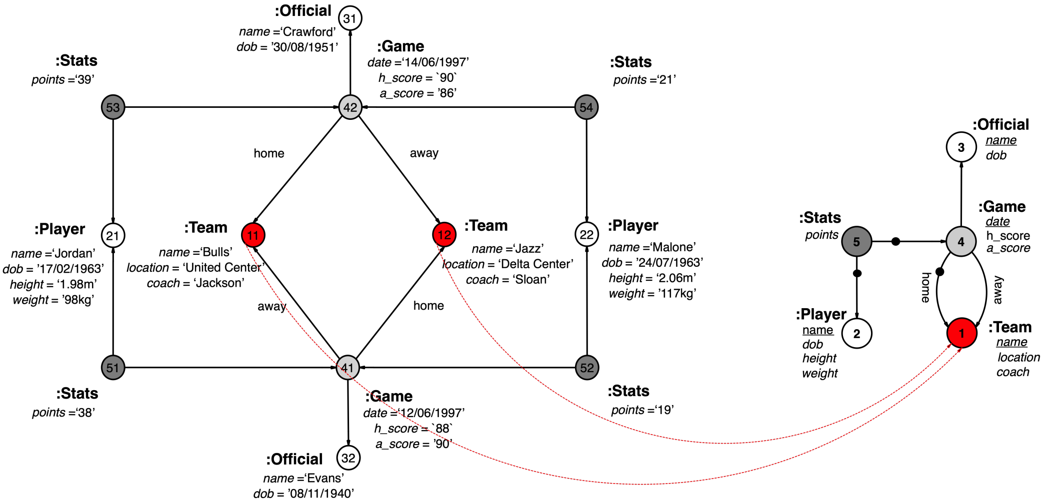




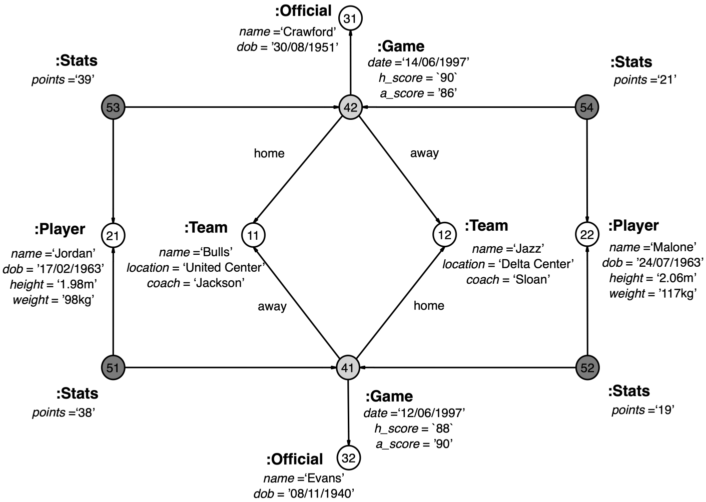
Homomorphism: Example of graph nodes mapped to node image

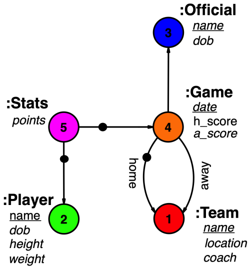


Homomorphism: Example of mapping graph nodes to node image

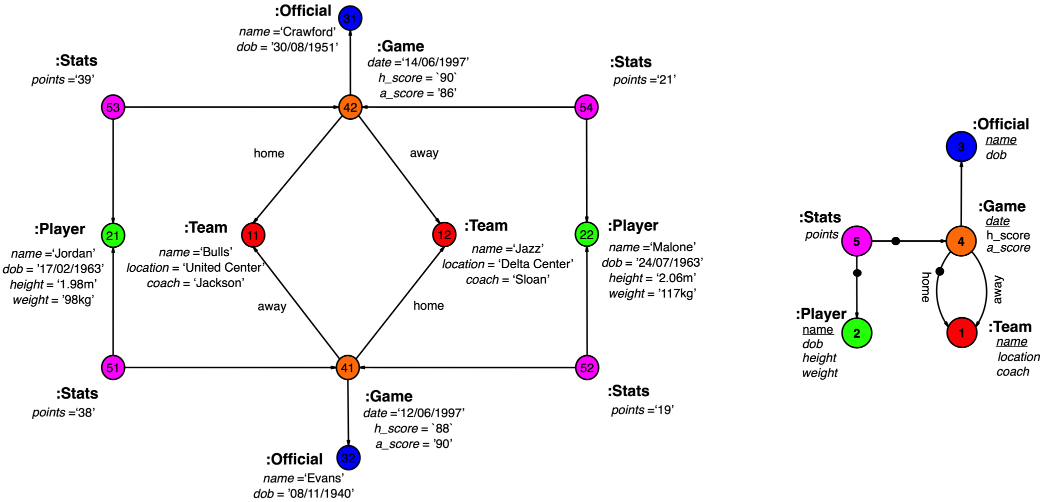


Homomorphism:Image for Node Mapping

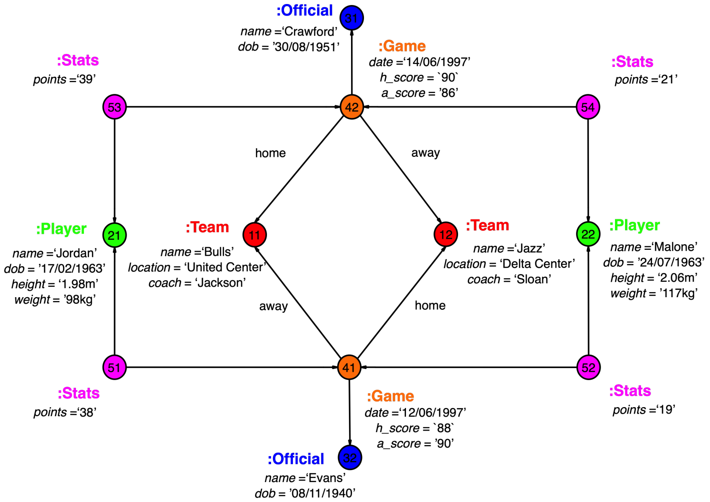


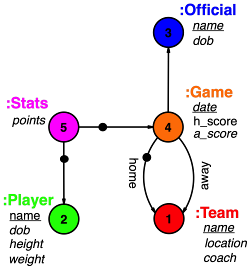


Homomorphism: Mapping of graph nodes to graph model nodes

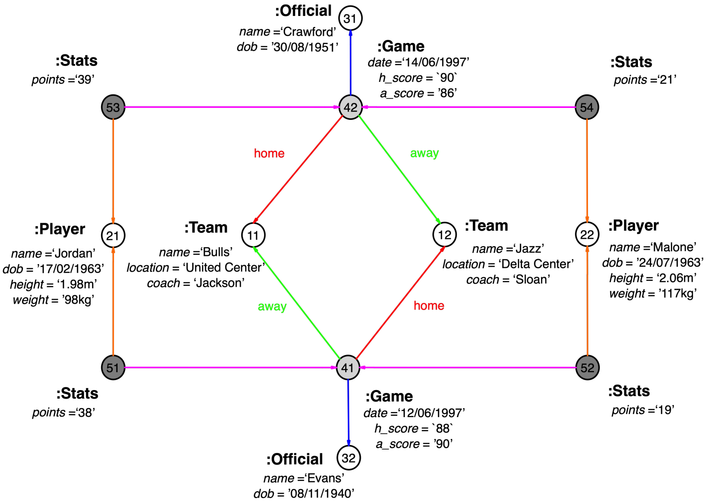


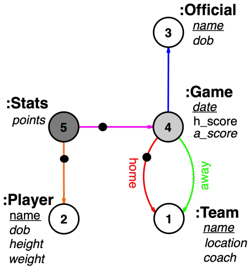
Homomorphism: Mapping of graph nodes to graph model nodes



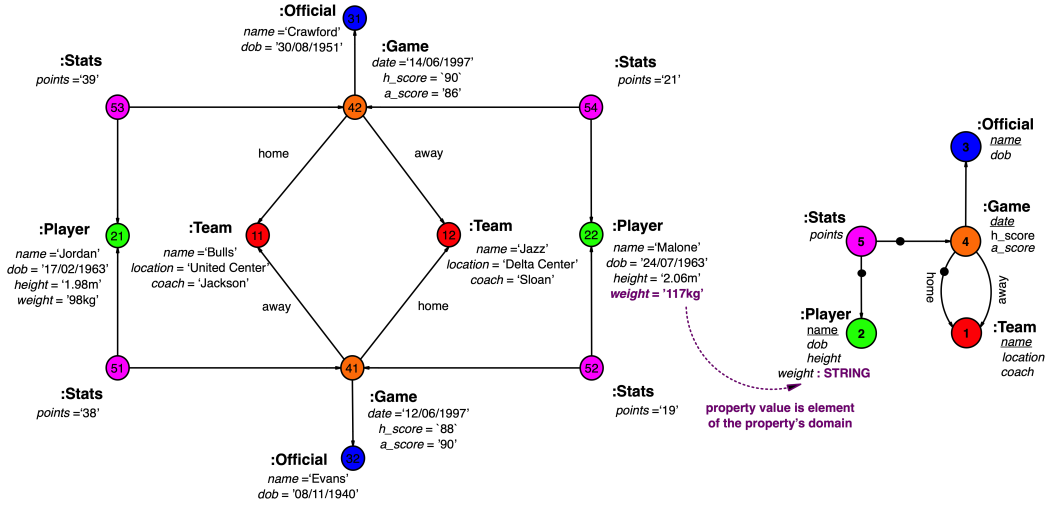


Homomorphism: Extension to edges

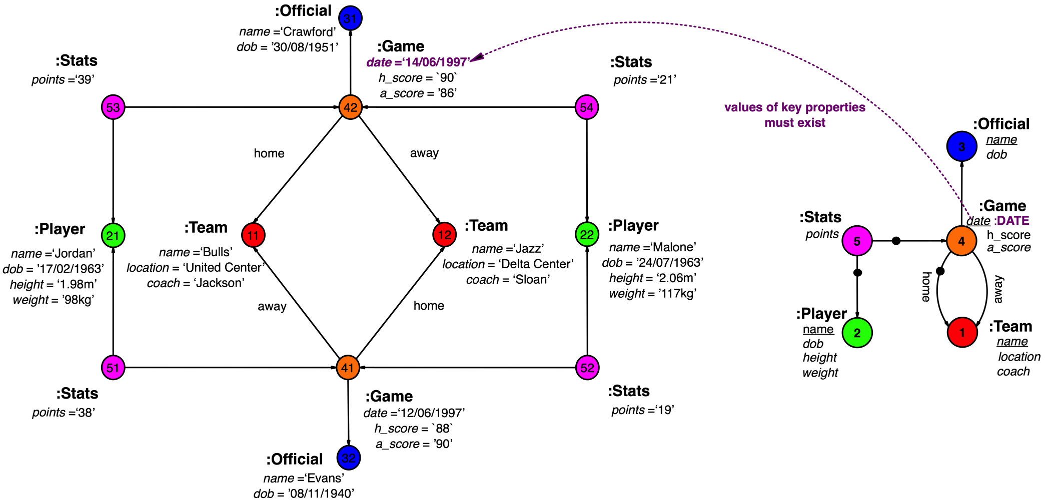




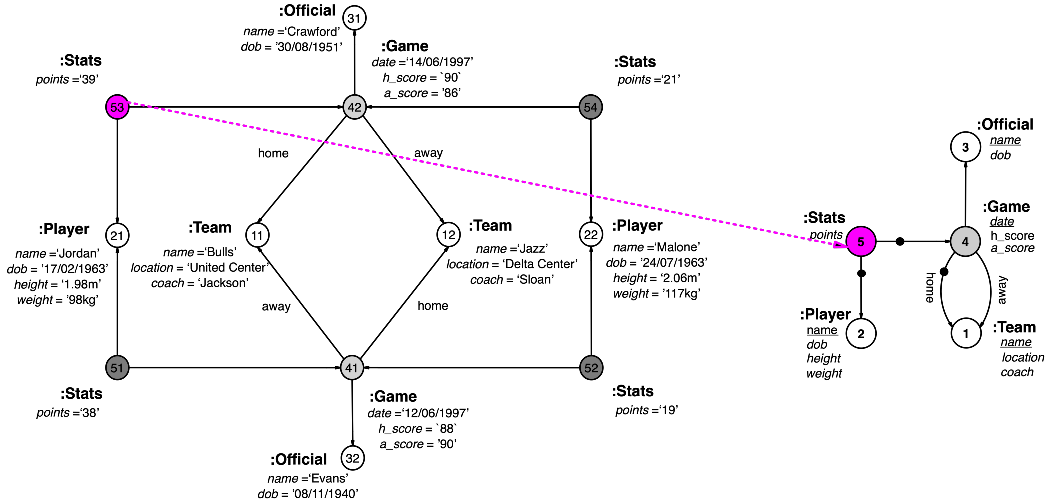
Homomorphism: Domain integrity



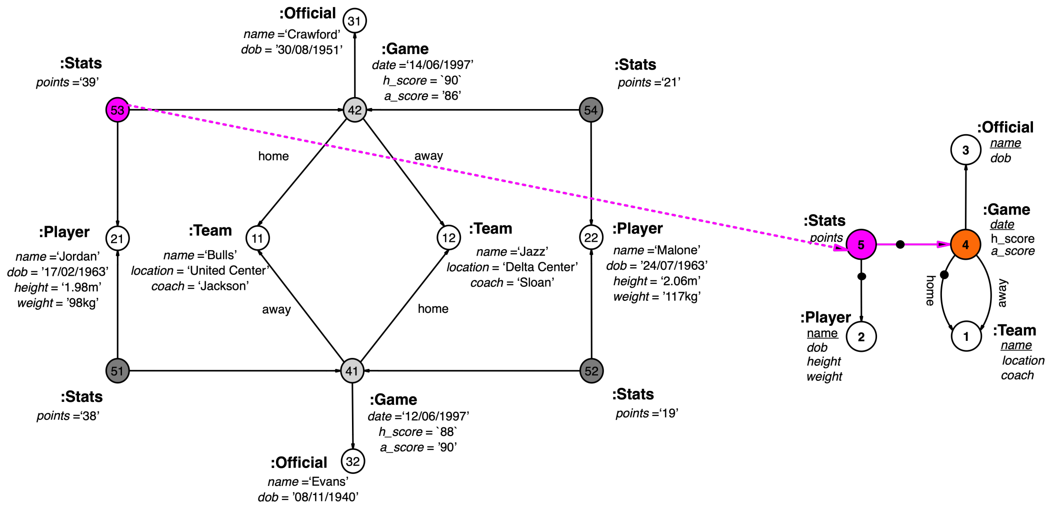
Homomorphism: Values of key properties must exist



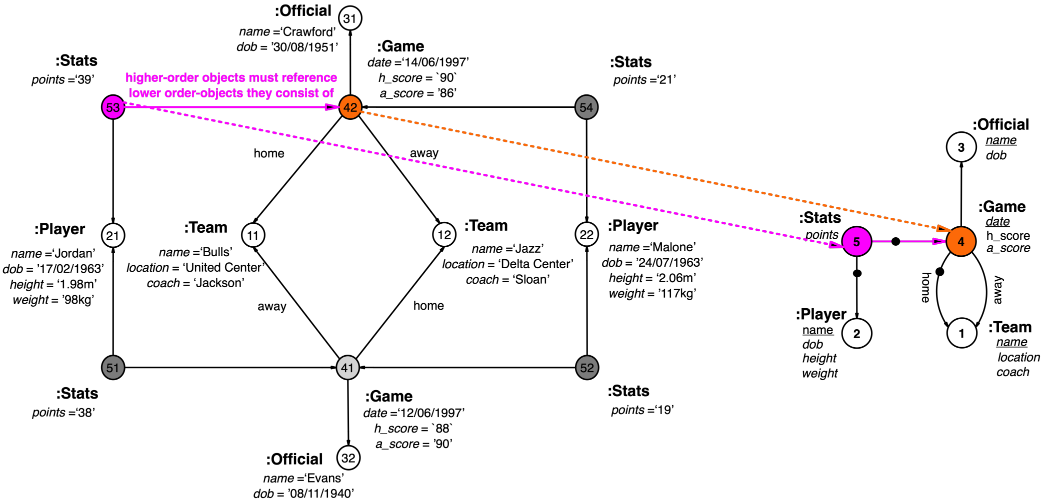
Homomorphism: Referential integrity for object



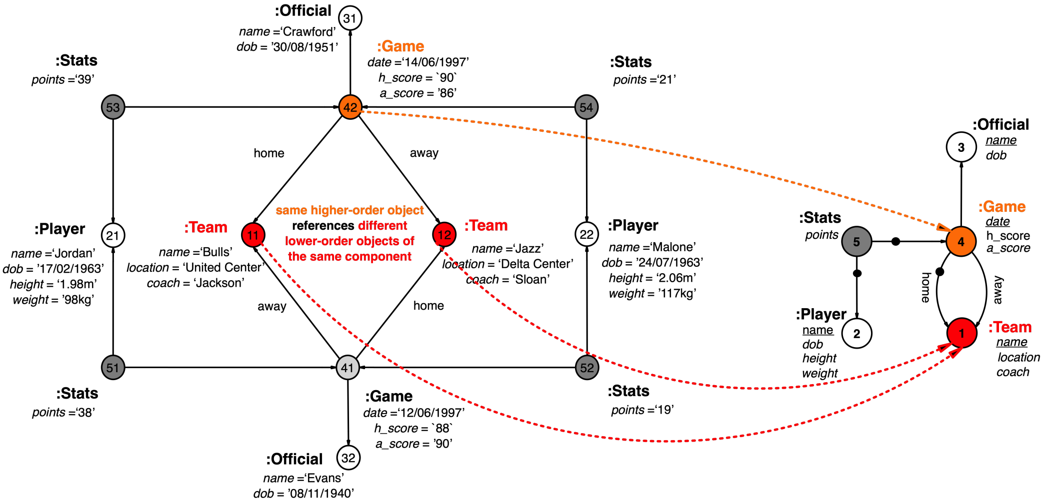
Homomorphism: Referential integrity for object of higher-order type



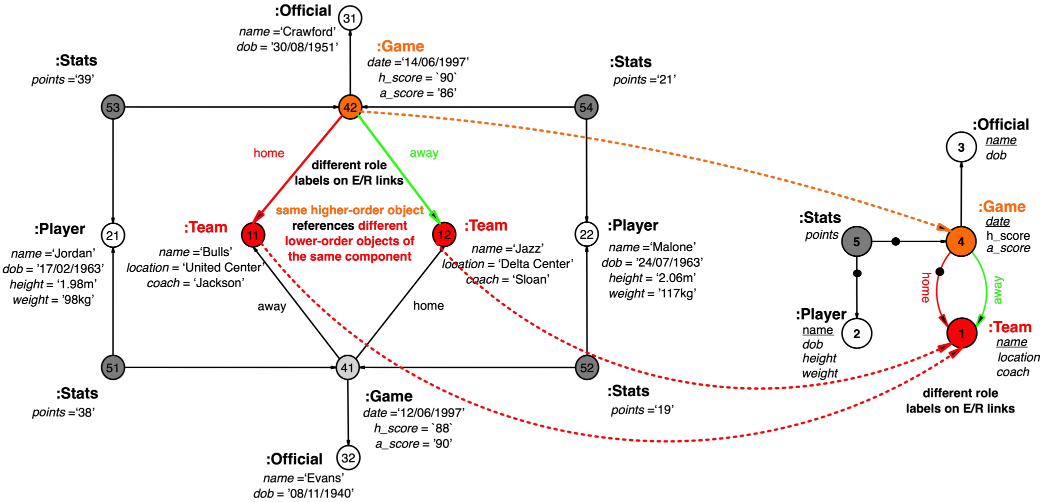
Homomorphism: Referential integrity for higher-order object with reference



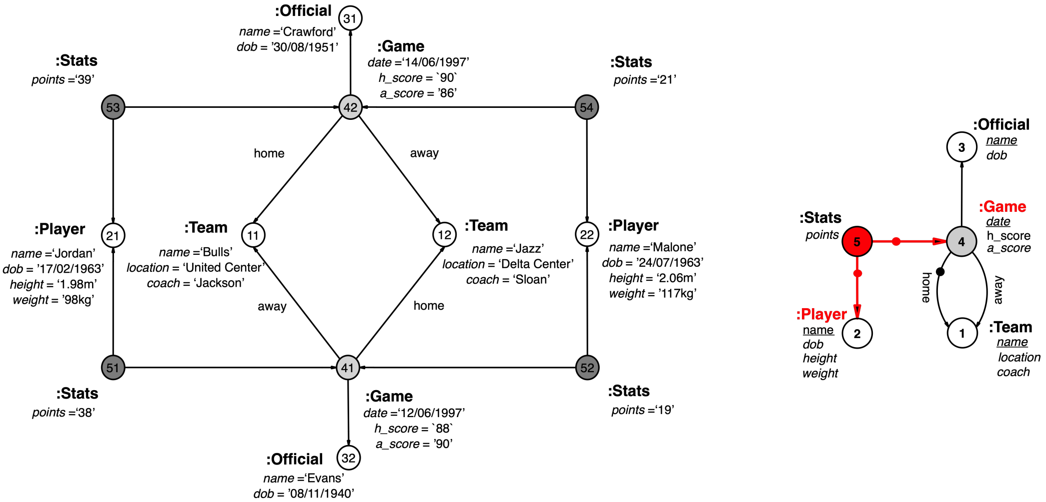
Homomorphism: Single-valued component



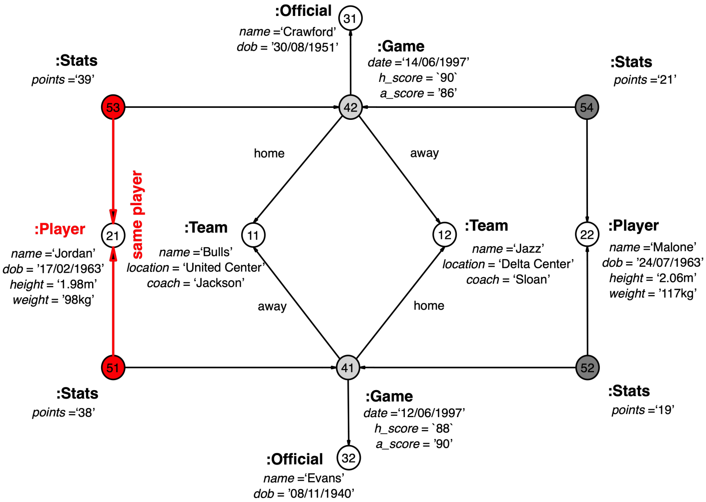
Homomorphism: Single-valued component distinguished by role labels

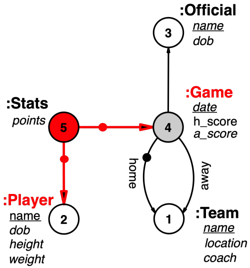


Homomorphism: Entity integrity on key $\textsc{Stats}(\{\textsc{Player}, \textsc{Game}\}, \emptyset)$

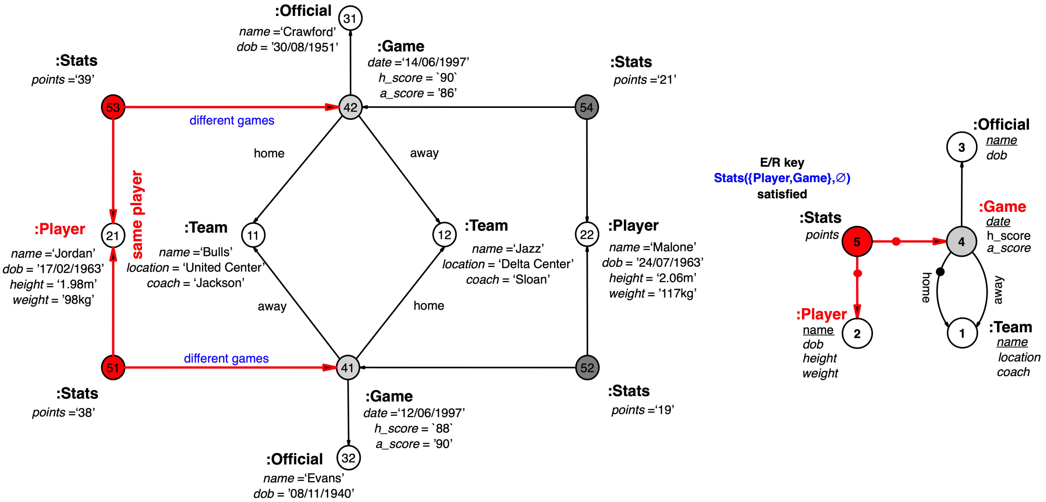


Homomorphism: Entity integrity on key $\textsc{Stats}(\{\textsc{Player}, \textsc{Game}\}, \emptyset)$

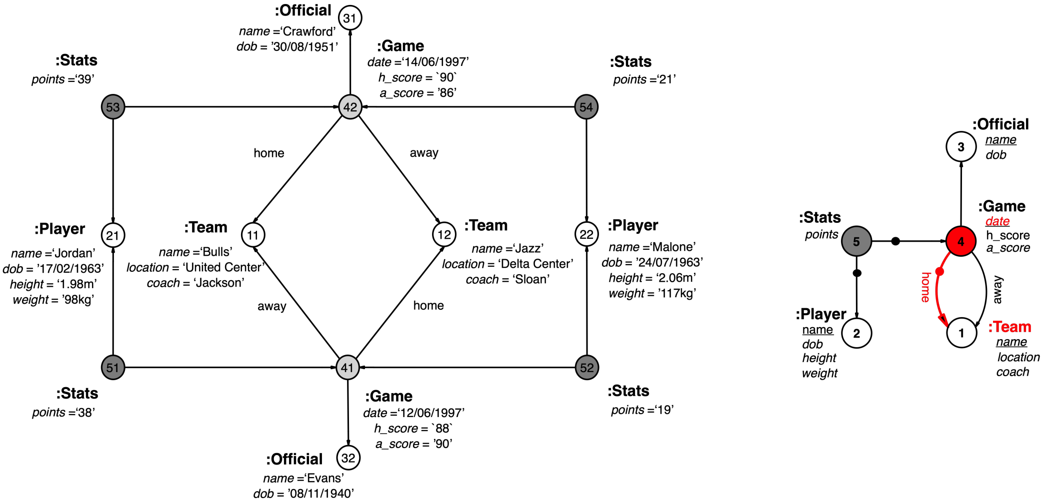




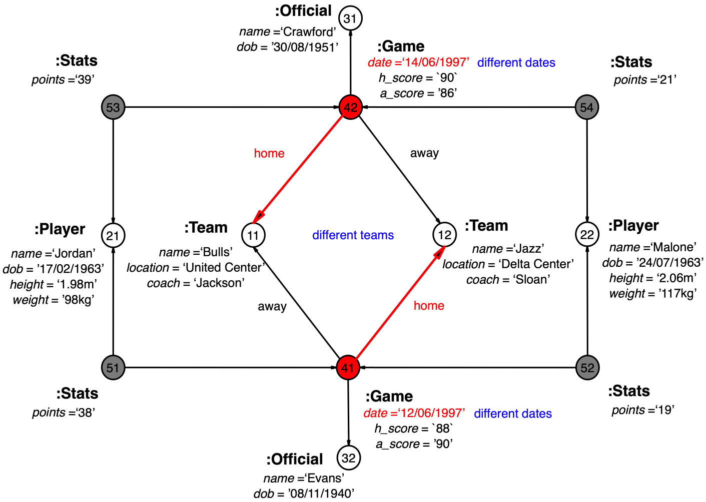
Homomorphism: Entity integrity on key STATS({PLAYER, GAME}, ∅)

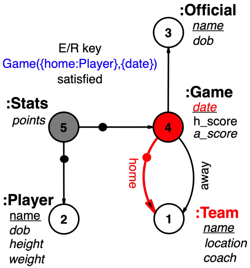


Homomorphism: Entity integrity on key GAME({*home*:TEAM}, {*date*})



Homomorphism: Entity integrity on key GAME({*home*:TEAM}, {*date*})





E/R graphs: Formal definition of E/R model graph compliance

E/R diagram $\mathcal{D} = (V, E)$ for E/R schema $\mathcal{S}$

- $\mathcal{O}_G$ as set of object identifiers,
- $\mathcal{L}_G \subseteq \mathcal{L}_D - \{\bullet\}$ as set of labels,
- $\mathcal{K}_G \subseteq \mathcal{K}_D$ as set of properties,
- $\mathcal{N}_G \subseteq \bigcup_{A \in attr(O), O \in \mathcal{S}} dom(A)$ as set of values

E/R graphs: Formal definition of E/R model graph compliance

A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $\mathcal{O}_G$, $\mathcal{L}_G$, $\mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ iff there is some function $h : \mathcal{O}_G \to \mathcal{O}_D$ such that:

E/R graphs: Formal definition of E/R model graph compliance

A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $\mathcal{O}_G$, $\mathcal{L}_G$, $\mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ iff there is some function $h : \mathcal{O}_G \to \mathcal{O}_D$ such that:

1. $h(V_G) \subseteq V_D$ and $h(E_G) \subseteq E_D$,

E/R graphs: Formal definition of E/R model graph compliance

A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $\mathcal{O}_G$, $\mathcal{L}_G$, $\mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ iff there is some function $h : \mathcal{O}_G \to \mathcal{O}_D$ such that:

1. $h(V_G) \subseteq V_D$ and $h(E_G) \subseteq E_D$,
2. $\forall o \in E_G\ (\eta_G(o) = (o_1, o_2) \Rightarrow h(\eta_G(o)) = (h(o_1), h(o_2)) = \eta_D(h(o)))$ (edge-preserving),

E/R graphs: Formal definition of E/R model graph compliance

A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $\mathcal{O}_G$, $\mathcal{L}_G$, $\mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ iff there is some function $h : \mathcal{O}_G \to \mathcal{O}_D$ such that:

1. $h(V_G) \subseteq V_D$ and $h(E_G) \subseteq E_D$,
2. $\forall o \in E_G \ (\eta_G(o) = (o_1, o_2) \Rightarrow h(\eta_G(o)) = (h(o_1), h(o_2)) = \eta_D(h(o)))$ (edge-preserving),
3. $\forall o \in V_G$, $\lambda_G(o) = \lambda_D(h(o))$ (node label-preserving),

E/R graphs: Formal definition of E/R model graph compliance

A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $\mathcal{O}_G$, $\mathcal{L}_G$, $\mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ iff there is some function $h : \mathcal{O}_G \to \mathcal{O}_D$ such that:

1. $h(V_G) \subseteq V_D$ and $h(E_G) \subseteq E_D$,
2. $\forall o \in E_G \ (\eta_G(o) = (o_1, o_2) \Rightarrow h(\eta_G(o)) = (h(o_1), h(o_2)) = \eta_D(h(o)))$ (edge-preserving),
3. $\forall o \in V_G$, $\lambda_G(o) = \lambda_D(h(o))$ (node label-preserving),
4. $\forall o \in E_G, l \neq \bullet$, $\lambda_G(o) = \{l\}$ if and only if $l \in \lambda_D(h(o))$ (role label-preserving),

E/R graphs: Formal definition of E/R model graph compliance

A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $\mathcal{O}_G$, $\mathcal{L}_G$, $\mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ iff there is some function $h : \mathcal{O}_G \to \mathcal{O}_D$ such that:

1. $h(V_G) \subseteq V_D$ and $h(E_G) \subseteq E_D$,
2. $\forall o \in E_G \; (\eta_G(o) = (o_1, o_2) \Rightarrow h(\eta_G(o)) = (h(o_1), h(o_2)) = \eta_D(h(o)))$ (edge-preserving),
3. $\forall o \in V_G$, $\lambda_G(o) = \lambda_D(h(o))$ (node label-preserving),
4. $\forall o \in E_G, l \neq \bullet$, $\lambda_G(o) = \{l\}$ if and only if $l \in \lambda_D(h(o))$ (role label-preserving),
5. $\forall v \in V_G, p \in \mathcal{K}_G \; (\nu_G(v, p) = val \Rightarrow val \in \nu_D(h(v), p))$ (type-preserving),

E/R graphs: Formal definition of E/R model graph compliance

A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $\mathcal{O}_G$, $\mathcal{L}_G$, $\mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ iff there is some function $h : \mathcal{O}_G \to \mathcal{O}_D$ such that:

1. $h(V_G) \subseteq V_D$ and $h(E_G) \subseteq E_D$,
2. $\forall o \in E_G \; (\eta_G(o) = (o_1, o_2) \Rightarrow h(\eta_G(o)) = (h(o_1), h(o_2)) = \eta_D(h(o)))$ (edge-preserving),
3. $\forall o \in V_G,\ \lambda_G(o) = \lambda_D(h(o))$ (node label-preserving),
4. $\forall o \in E_G, l \neq \bullet,\ \lambda_G(o) = \{l\}$ if and only if $l \in \lambda_D(h(o))$ (role label-preserving),
5. $\forall v \in V_G, p \in \mathcal{K}_G \; (\nu_G(v, p) = val \Rightarrow val \in \nu_D(h(v), p))$ (type-preserving),
6. $\forall v \in V_G, p \in \mathcal{K}_D \; (\nu_D(h(v), p) = dom(p) \Rightarrow \exists val \in dom(p) \; (\nu_G(v, p) = val))$ (key properties must exist),

E/R graphs: Formal definition of E/R model graph compliance

A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $\mathcal{O}_G$, $\mathcal{L}_G$, $\mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ iff there is some function $h : \mathcal{O}_G \rightarrow \mathcal{O}_D$ such that:

1. $h(V_G) \subseteq V_D$ and $h(E_G) \subseteq E_D$,
2. $\forall o \in E_G \; (\eta_G(o) = (o_1, o_2) \Rightarrow h(\eta_G(o)) = (h(o_1), h(o_2)) = \eta_D(h(o)))$ (edge-preserving),
3. $\forall o \in V_G, \; \lambda_G(o) = \lambda_D(h(o))$ (node label-preserving),
4. $\forall o \in E_G, l \neq \bullet, \; \lambda_G(o) = \{l\}$ if and only if $l \in \lambda_D(h(o))$ (role label-preserving),
5. $\forall v \in V_G, p \in \mathcal{K}_G \; (\nu_G(v, p) = val \Rightarrow val \in \nu_D(h(v), p))$ (type-preserving),
6. $\forall v \in V_G, p \in \mathcal{K}_D \; (\nu_D(h(v), p) = dom(p) \Rightarrow \exists val \in dom(p) \; (\nu_G(v, p) = val))$

   (key properties must exist),
7. $\forall o, o' \in E_G \; ((\eta_G(o) = (u, v), \eta_G(o') = (u, w) \in E_G \wedge \lambda_G(v) = \lambda_G(w) \wedge \lambda_G(o) = \lambda_G(o'))) \Rightarrow v = w$
   (single-valued components, that is, equal target labels and role labels mean equal target nodes),

E/R graphs: Formal definition of E/R model graph compliance

A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $\mathcal{O}_G$, $\mathcal{L}_G$, $\mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ iff there is some function $h : \mathcal{O}_G \to \mathcal{O}_D$ such that:

1. $h(V_G) \subseteq V_D$ and $h(E_G) \subseteq E_D$,
2. $\forall o \in E_G \; (\eta_G(o) = (o_1, o_2) \Rightarrow h(\eta_G(o)) = (h(o_1), h(o_2)) = \eta_D(h(o)))$ (edge-preserving),
3. $\forall o \in V_G,\ \lambda_G(o) = \lambda_D(h(o))$ (node label-preserving),
4. $\forall o \in E_G, l \neq \bullet,\ \lambda_G(o) = \{l\}$ if and only if $l \in \lambda_D(h(o))$ (role label-preserving),
5. $\forall v \in V_G, p \in \mathcal{K}_G \, (\nu_G(v, p) = val \Rightarrow val \in \nu_D(h(v), p))$ (type-preserving),
6. $\forall v \in V_G, p \in \mathcal{K}_D \, (\nu_D(h(v), p) = dom(p) \Rightarrow \exists val \in dom(p) \, (\nu_G(v, p) = val))$ (key properties must exist),
7. $\forall o, o' \in E_G \, ((\eta_G(o) = (u, v), \eta_G(o') = (u, w) \in E_G \wedge \lambda_G(v) = \lambda_G(w) \wedge \lambda_G(o) = \lambda_G(o'))) \Rightarrow v = w$ (single-valued components, that is, equal target labels and role labels mean equal target nodes),
8. $\forall v \in V_G, w' \in V_D \, ((h(v), w') \in E_D \Rightarrow \exists w \in V_G \, ((v, w) \in E_G \wedge h(w) = w'))$ (referential integrity), and

E/R graphs: Formal definition of E/R model graph compliance

A property graph $G = (V_G, E_G, \eta_G, \lambda_G, \nu_G)$ over $\mathcal{O}_G$, $\mathcal{L}_G$, $\mathcal{K}_G$, and $\mathcal{N}_G$ is called an *E/R graph* for $\mathcal{G}_D$ iff there is some function $h : \mathcal{O}_G \to \mathcal{O}_D$ such that:

1. $h(V_G) \subseteq V_D$ and $h(E_G) \subseteq E_D$,
2. $\forall o \in E_G \; (\eta_G(o) = (o_1, o_2) \Rightarrow h(\eta_G(o)) = (h(o_1), h(o_2)) = \eta_D(h(o)))$ (edge-preserving),
3. $\forall o \in V_G$, $\lambda_G(o) = \lambda_D(h(o))$ (node label-preserving),
4. $\forall o \in E_G, l \neq \bullet$, $\lambda_G(o) = \{l\}$ if and only if $l \in \lambda_D(h(o))$ (role label-preserving),
5. $\forall v \in V_G, p \in \mathcal{K}_G \; (\nu_G(v, p) = val \Rightarrow val \in \nu_D(h(v), p))$ (type-preserving),
6. $\forall v \in V_G, p \in \mathcal{K}_D \; (\nu_D(h(v), p) = dom(p) \Rightarrow \exists val \in dom(p) \, (\nu_G(v, p) = val))$ (key properties must exist),
7. $\forall o, o' \in E_G \; ((\eta_G(o) = (u, v), \eta_G(o') = (u, w) \in E_G \wedge \lambda_G(v) = \lambda_G(w) \wedge \lambda_G(o) = \lambda_G(o'))) \Rightarrow v = w$ (single-valued components, that is, equal target labels and role labels mean equal target nodes),
8. $\forall v \in V_G, w' \in V_D \; ((h(v), w') \in E_D \Rightarrow \exists w \in V_G \; ((v, w) \in E_G \wedge h(w) = w'))$ (referential integrity), and
9. $\forall O \in V_D$ such that $id(O) = C \cup K$ with $C = \{O_1, \ldots, O_n\}$ and $K = \{K_1, \ldots, K_m\}$, G satisfies the E/R key $O(C, K)$ (entity integrity).

E/R keys

E/R keys for an E/R graph model $\mathcal{G}_D$ of an E/R diagram $\mathcal{D}$

- For every object type $O \in V_D$, it is an expression $O(C, K)$ with
  - $C = \{O_1, \ldots, O_n\} \subseteq \mathit{comp}(O)$ [key components]
  - $K = \{K_1, \ldots, K_m\} \subseteq \mathit{attr}(O)$ [key properties]

E/R keys

E/R keys for an E/R graph model $\mathcal{G}_D$ of an E/R diagram $\mathcal{D}$

- For every object type $O \in V_D$, it is an expression $O(C, K)$ with
  - $C = \{O_1, \ldots, O_n\} \subseteq comp(O)$ [key components]
  - $K = \{K_1, \ldots, K_m\} \subseteq attr(O)$ [key properties]
- An *E/R graph* G for $\mathcal{G}_D$ *satisfies* $O(C, K)$ iff for all $o, o' \in V_G$ such that
  - $(o, o_i), (o', o_i) \in E_G$ for $i = 1, \ldots, n$,
  - $\downarrow = \nu_G(o, K_j) = \nu_G(o', K_j) = \downarrow$ for $j = 1, \ldots, m$,
  - $\lambda_G(o) = O = \lambda_G(o')$, and $\lambda_G(o_i) = O_i$ for $i = 1, \ldots, n$,

  then $o = o'$.

E/R keys

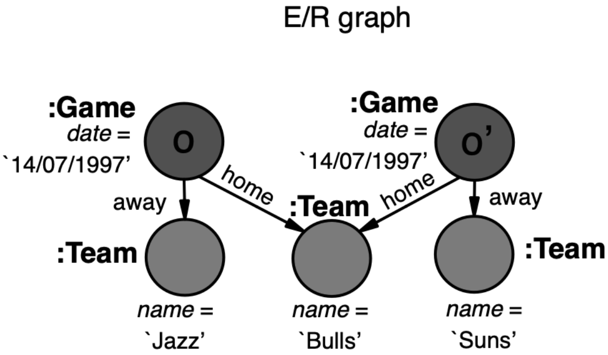
E/R keys for an E/R graph model $\mathcal{G}_D$ of an E/R diagram $\mathcal{D}$

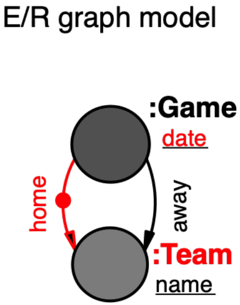
- For every object type $O \in V_D$, it is an expression $O(C, K)$ with
  - $C = \{O_1, \ldots, O_n\} \subseteq comp(O)$ [key components]
  - $K = \{K_1, \ldots, K_m\} \subseteq attr(O)$ [key properties]
- An *E/R graph* G for $\mathcal{G}_D$ *satisfies* $O(C, K)$ iff for all $o, o' \in V_G$ such that
  - $(o, o_i), (o', o_i) \in E_G$ for $i = 1, \ldots, n$,
  - $\downarrow = \nu_G(o, K_j) = \nu_G(o', K_j) = \downarrow$ for $j = 1, \ldots, m$,
  - $\lambda_G(o) = O = \lambda_G(o')$, and $\lambda_G(o_i) = O_i$ for $i = 1, \ldots, n$,

  then $o = o'$.

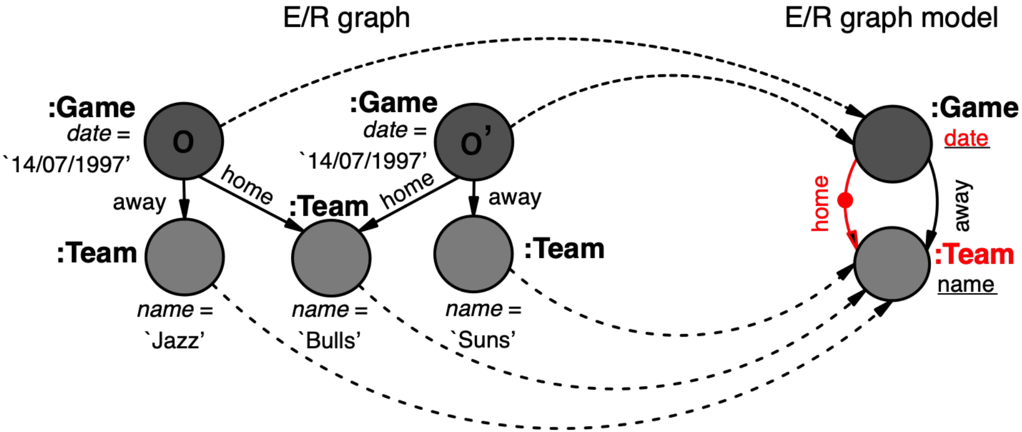
- E/R keys $E(C, K)$ where $C = \emptyset$ are called *property keys*
- E/R keys do not stipulate object uniqueness if any key property is undefined
- Principle of entity integrity requires some E/R key for each object type
  - ensures objects can be identified uniquely and accessed efficiently
- We recommend specifying all E/R keys that express integrity rules

E/R Key Example $\textsc{Game}(\{\textit{home}{:}\textsc{Team}\}, \{\textit{date}\})$

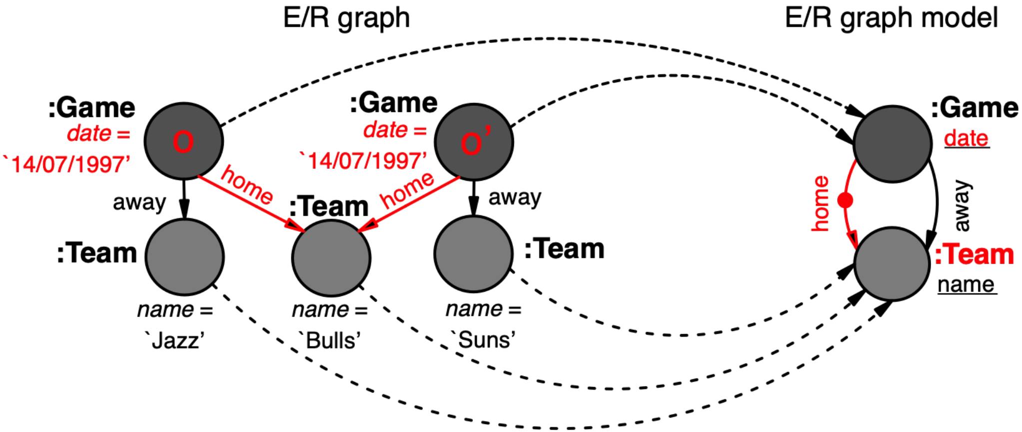




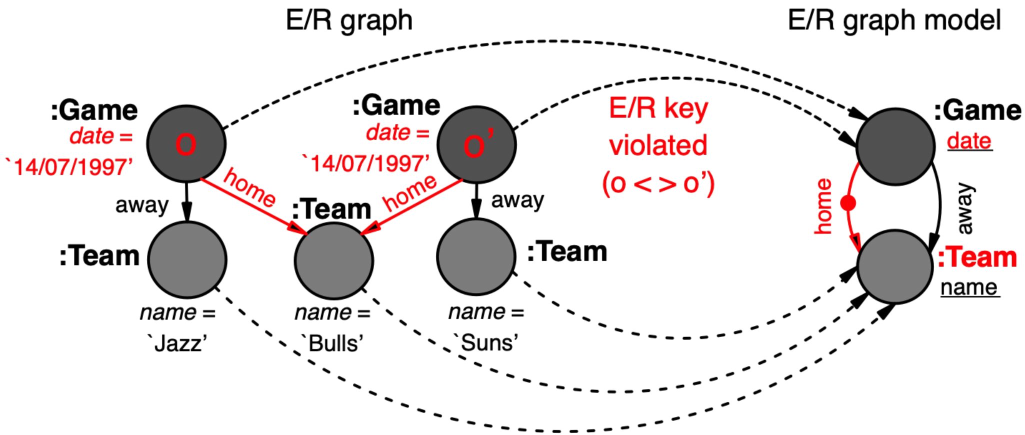
E/R Key Example $\text{GAME}(\{\mathit{home}{:}\text{TEAM}\}, \{\mathit{date}\})$



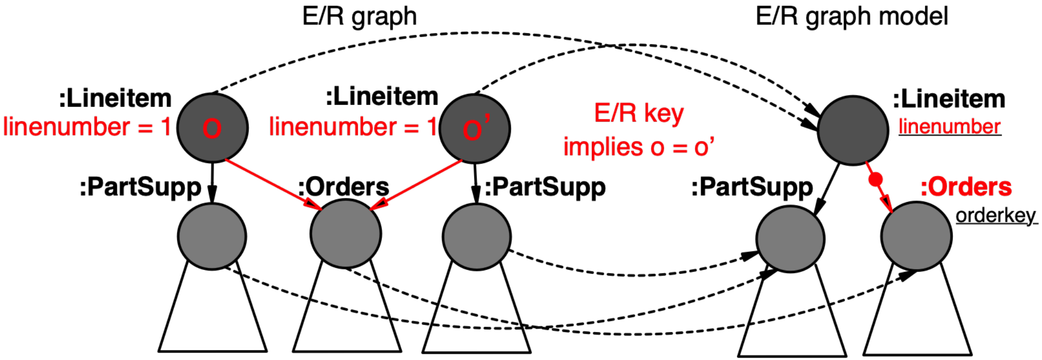
E/R Key Example $\textsc{Game}(\{\textit{home}{:}\textsc{Team}\}, \{\textit{date}\})$



E/R Key Example GAME({*home*:TEAM}, {*date*})



Another E/R Key Example $\textsc{Lineitem}(\{\textsc{Orders}\}, \{\textit{linenumber}\})$



Pitstop 2

Contributions

- E/R Graphs are natural instances of E/R graph models, defining homomorphisms
- E/R Graphs provide a graph semantics for E/R schemata and diagrams

Pitstop 2

Contributions

- E/R Graphs are natural instances of E/R graph models, defining homomorphisms
- E/R Graphs provide a graph semantics for E/R schemata and diagrams

E/R Graphs: A Natural Semantics for Well-designed Databases

Entity and Referential Integrity

PG-Key

When specifying some set of PG-keys, what other PG-keys do you specify implicitly?

PG-Key

When specifying some set of PG-keys, what other PG-keys do you specify implicitly?

Relational translation of relational database into property graph

- $(S, \Sigma_{\mathcal{S}})$: finite set of keys and foreign keys over relational database schema S
- Translate into $\mathcal{G}_{\mathcal{S}}$ with a set $\Sigma_{\mathcal{G}_{\mathcal{S}}} = \{\sigma_{\mathcal{G}_{\mathcal{S}}} \mid \sigma \in \Sigma_{\mathcal{S}}\}$ of PG-keys by mapping
  - every relation schema $R \in \mathcal{S}$ to a vertex v_R with label $:R$,

PG-Key

When specifying some set of PG-keys, what other PG-keys do you specify implicitly?

Relational translation of relational database into property graph

- $(S, \Sigma_{\mathcal{S}})$: finite set of keys and foreign keys over relational database schema S
- Translate into $\mathcal{G}_{\mathcal{S}}$ with a set $\Sigma_{\mathcal{G}_{\mathcal{S}}} = \{\sigma_{\mathcal{G}_{\mathcal{S}}} \mid \sigma \in \Sigma_{\mathcal{S}}\}$ of PG-keys by mapping
  - every relation schema $R \in \mathcal{S}$ to a vertex v_R with label $:R$,
  - every attribute $A \in R$ to a property A on the node v_R,

PG-Key

When specifying some set of PG-keys, what other PG-keys do you specify implicitly?

Relational translation of relational database into property graph

- (S, Σ_S): finite set of keys and foreign keys over relational database schema S
- Translate into $\mathcal{G}_S$ with a set $\Sigma_{\mathcal{G}_S} = \{\sigma_{\mathcal{G}_S} \mid \sigma \in \Sigma_S\}$ of PG-keys by mapping
  - every relation schema $R \in \mathcal{S}$ to a vertex v_R with label $:R$,
  - every attribute $A \in R$ to a property A on the node v_R,
  - every key $\sigma = \{A_1, \ldots, A_n\}$ over R to a PG-key $\sigma_{\mathcal{G}_S}$ over v_R:
    For x:R IDENTIFIER $x.A_1, \ldots, x.A_n$

PG-Key

When specifying some set of PG-keys, what other PG-keys do you specify implicitly?

Relational translation of relational database into property graph

- $(S, \Sigma_{\mathcal{S}})$: finite set of keys and foreign keys over relational database schema S
- Translate into $\mathcal{G}_{\mathcal{S}}$ with a set $\Sigma_{\mathcal{G}_{\mathcal{S}}} = \{\sigma_{\mathcal{G}_{\mathcal{S}}} \mid \sigma \in \Sigma_{\mathcal{S}}\}$ of PG-keys by mapping
  - every relation schema $R \in \mathcal{S}$ to a vertex v_R with label $:R$,
  - every attribute $A \in R$ to a property A on the node v_R,
  - every key $\sigma = \{A_1, \ldots, A_n\}$ over R to a PG-key $\sigma_{\mathcal{G}_{\mathcal{S}}}$ over v_R:
    For x:R IDENTIFIER $x.A_1, \ldots, x.A_n$
  - every foreign key $\sigma = R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$ to a PG-key $\sigma_{\mathcal{G}_{\mathcal{S}}}$ over v_R:
    For x:R IDENTIFIER $x.A_1, \ldots, x.A_n, y$ WITHIN $(x), (y{:}S)$ WHERE
    $x.A_1 = y.B_1, \ldots, x.A_n = y.B_n$.

PG-Key

When specifying some set of PG-keys, what other PG-keys do you specify implicitly?

Relational translation of relational database into property graph

- (S, Σ_S): finite set of keys and foreign keys over relational database schema S
- Translate into $\mathcal{G}_S$ with a set $\Sigma_{\mathcal{G}_S} = \{\sigma_{\mathcal{G}_S} \mid \sigma \in \Sigma_S\}$ of PG-keys by mapping
  - every relation schema $R \in \mathcal{S}$ to a vertex v_R with label $:R$,
  - every attribute $A \in R$ to a property A on the node v_R,
  - every key $\sigma = \{A_1, \ldots, A_n\}$ over R to a PG-key $\sigma_{\mathcal{G}_S}$ over v_R:
    For $x{:}R$ IDENTIFIER $x.A_1, \ldots, x.A_n$
  - every foreign key $\sigma = R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$ to a PG-key $\sigma_{\mathcal{G}_S}$ over v_R:
    For $x{:}R$ IDENTIFIER $x.A_1, \ldots, x.A_n, y$ WITHIN $(x), (y{:}S)$ WHERE
    $x.A_1 = y.B_1, \ldots, x.A_n = y.B_n$.
- Extend mapping to relational databases over (S, Σ_S) by translating every record r over schema R to a node v_r with label $:R$ and the property-value pairs $A = r(A)$ for every $A \in R$

Reasoning about relational PG-Keys is infeasible

The *implication problem for relational PG-keys*

Decide whether for every given

$$(\mathcal{S}, \Sigma_{\mathcal{S}} \cup \{\varphi\}),$$

every instance $\mathcal{I}(\mathcal{G}_{\mathcal{S}})$ over $\mathcal{G}_{\mathcal{S}}$ that satisfies all elements of $\Sigma_{\mathcal{G}_{\mathcal{S}}}$ will also satisfy $\varphi_{\mathcal{G}_{\mathcal{S}}}$

Reasoning about relational PG-Keys is infeasible

The *implication problem for relational PG-keys*

Decide whether for every given

$$(\mathcal{S}, \Sigma_{\mathcal{S}} \cup \{\varphi\}),$$

every instance $\mathcal{I}(\mathcal{G}_{\mathcal{S}})$ over $\mathcal{G}_{\mathcal{S}}$ that satisfies all elements of $\Sigma_{\mathcal{G}_{\mathcal{S}}}$ will also satisfy $\varphi_{\mathcal{G}_{\mathcal{S}}}$

Theorem

The implication problem of relational PG-keys is undecidable.

E/R Keys and their Implication Problem

E/R Keys as PG-Keys

E/R key $O(\{O_1, \ldots, O_n\}, \{K_1, \ldots, K_m\})$ for $\mathcal{G}_D$ is satisfied by an E/R graph G for $\mathcal{G}_D$ if and only if G satisfies the following PG-key

$$\text{FOR } (x{:}O) \text{ IDENTIFIER } x.K_1, \ldots, x.K_m, y_1, \ldots, y_n \text{ WITHIN}$$
$$(x) \to (y_1{:}O_1), \ldots, (x) \to (y_n{:}O_n).$$

E/R Keys and their Implication Problem

E/R Keys as PG-Keys

E/R key $O(\{O_1, \ldots, O_n\}, \{K_1, \ldots, K_m\})$ for $\mathcal{G}_D$ is satisfied by an E/R graph G for $\mathcal{G}_D$ if and only if G satisfies the following PG-key

$$\text{FOR } (x{:}O) \text{ IDENTIFIER } x.K_1, \ldots, x.K_m, y_1, \ldots, y_n \text{ WITHIN}$$
$$(x) \to (y_1{:}O_1), \ldots, (x) \to (y_n{:}O_n).$$

Implication problem for E/R keys

Decide whether for every E/R graph model $\mathcal{G}_D$ and every set $\Sigma \cup \{\varphi\}$ of E/R keys for $\mathcal{G}_D$, every E/R graph that satisfies all E/R keys in Σ also satisfies φ.

Efficient Reasoning about E/R Keys

Theorem (Axiomatic Characterization)

The implication of E/R keys is finitely axiomatized by the extension rule:

$$\frac{O(C,K)}{O(CC',KK')}$$

Efficient Reasoning about E/R Keys

Theorem (Axiomatic Characterization)

The implication of E/R keys is finitely axiomatized by the extension rule:

$$\frac{O(C,K)}{O(CC',KK')}$$

Σ implies $O(C,K)$

if and only if

there is some E/R key $O(C',K')$ in Σ such that $C' \subseteq C$ and $K' \subseteq K$.

Efficient Reasoning about E/R Keys

Theorem (Axiomatic Characterization)

The implication of E/R keys is finitely axiomatized by the extension rule:

$$\frac{O(C,K)}{O(CC',KK')}$$

Σ implies $O(C,K)$
if and only if
there is some E/R key $O(C',K')$ in Σ such that $C' \subseteq C$ and $K' \subseteq K$.

Corollary (Algorithmic Characterization)

The implication of E/R keys is decidable in time linear in the input.

Pitstop 3

E/R keys

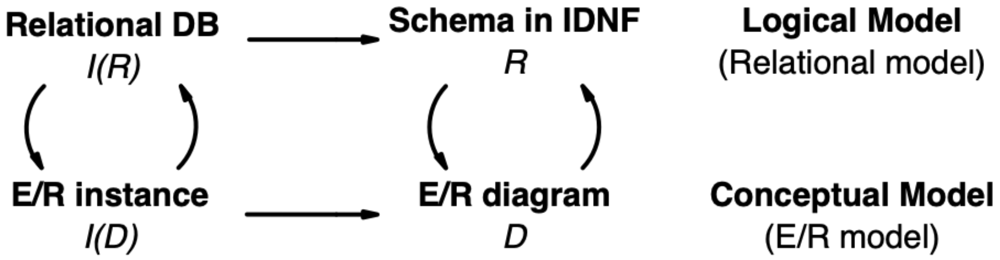
- form a computationally-attractive fragment of PG-Key for managing entity and referential integrity of well-designed property graphs

E/R graphs and their models
unify
conceptual, logical and graph modeling

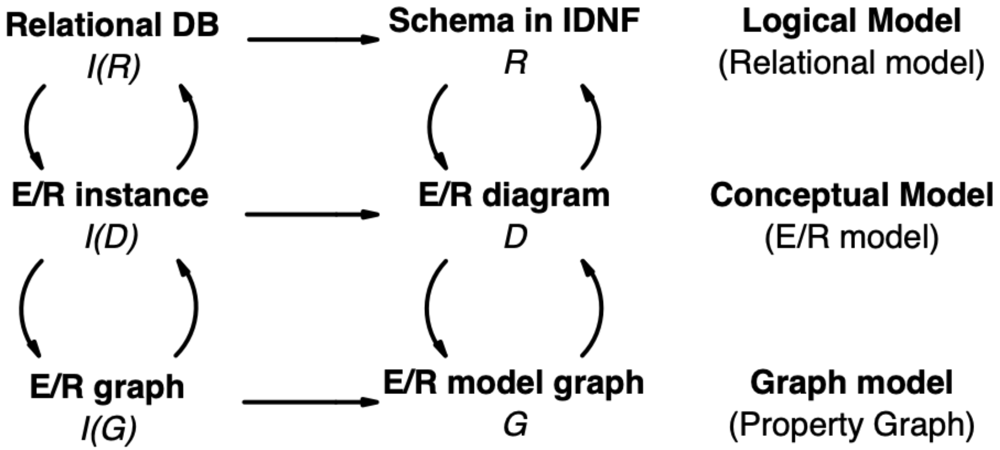
Well-designed Databases: Conceptual Perspective

E/R instance	$\longrightarrow$	**E/R diagram**	**Conceptual Model**
I(D)		*D*	(E/R model)

Well-designed Databases: Conceptual, Logical Perspective



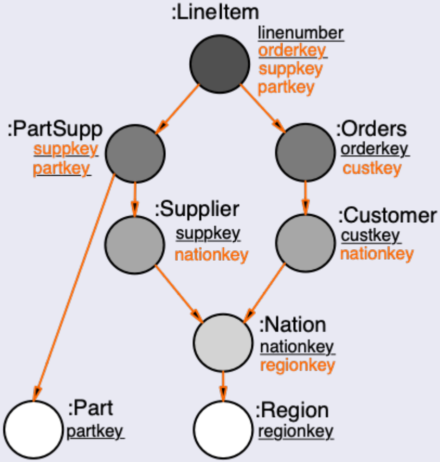
Well-designed Databases: Conceptual, Logical and Graph Perspective



Taking the Management of Entity and Referential Integrity Management to the Next Level

Integrity for Property Graphs: Spoilt for Choice

E/R graph model

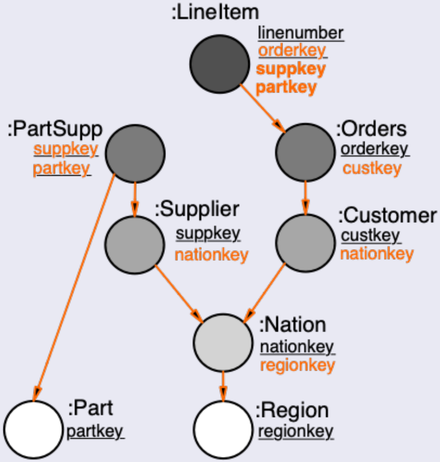


E/R Keys

- Each E/R link provides a choice:
  - duplicate key properties of target node on source node, or
  - keep the E/R link
- for the first choice:
  - E/R links are redundant
  - E/R keys reduce to property keys
- for the second choice:
  - no property redundancy
  - E/R keys may require key components

Integrity for Property Graphs: Spoilt for Choice

E/R graph model

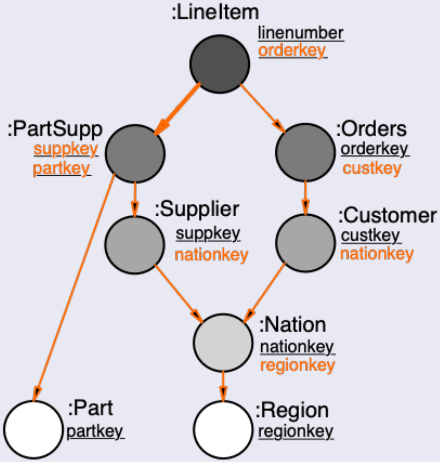


E/R Keys

- Each E/R link provides a choice:
  - duplicate key properties of target node on source node, or
  - keep the E/R link
- for the first choice:
  - E/R links are redundant
  - E/R keys reduce to property keys
- for the second choice:
  - no property redundancy
  - E/R keys may require key components

Integrity for Property Graphs: Spoilt for Choice

E/R graph model

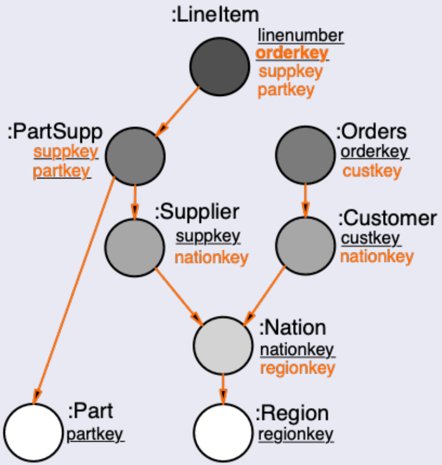


E/R Keys

- Each E/R link provides a choice:
  - duplicate key properties of target node on source node, or
  - keep the E/R link
- for the first choice:
  - E/R links are redundant
  - E/R keys reduce to property keys
- for the second choice:
  - no property redundancy
  - E/R keys may require key components

Integrity for Property Graphs: Spoilt for Choice

E/R graph model

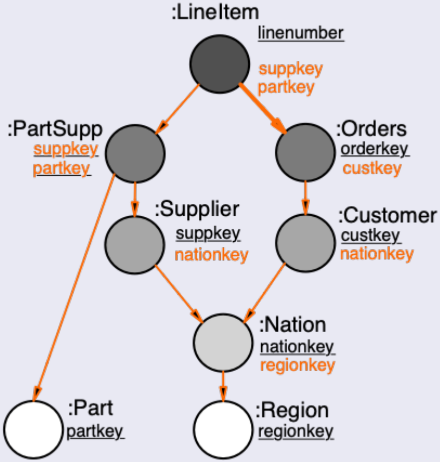


E/R Keys

- Each E/R link provides a choice:
  - duplicate key properties of target node on source node, or
  - keep the E/R link
- for the first choice:
  - E/R links are redundant
  - E/R keys reduce to property keys
- for the second choice:
  - no property redundancy
  - E/R keys may require key components

Integrity for Property Graphs: Spoilt for Choice

E/R graph model

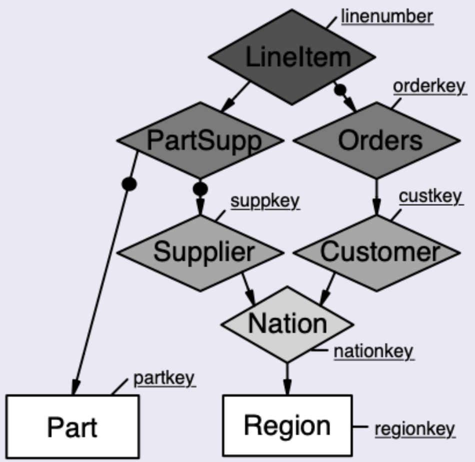


E/R Keys

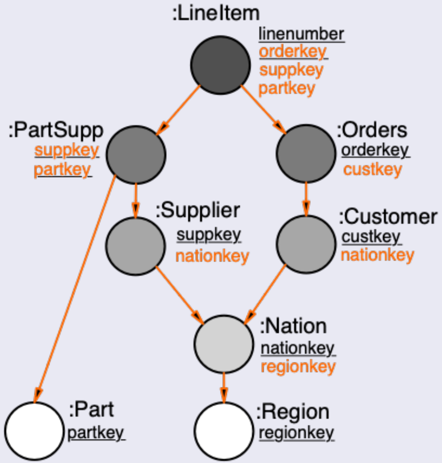
- Each E/R link provides a choice:
  - duplicate key properties of target node on source node, or
  - keep the E/R link
- for the first choice:
  - E/R links are redundant
  - E/R keys reduce to property keys
- for the second choice:
  - no property redundancy
  - E/R keys may require key components

Relational semantics: duplicate key properties and use property keys

E/R graph model

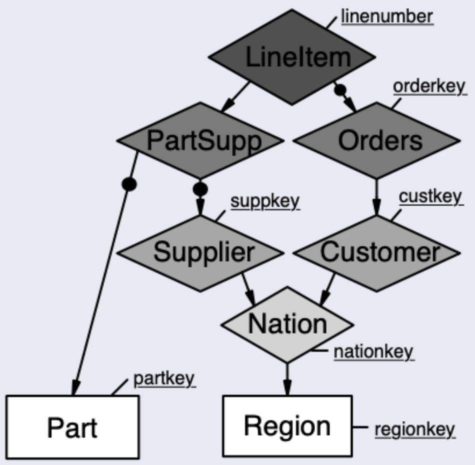


Relational semantics

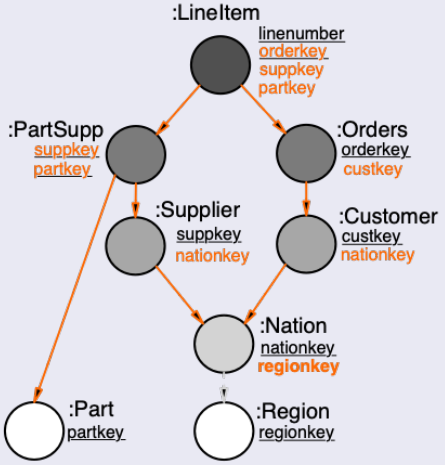


Relational semantics: duplicate key properties and use property keys

E/R graph model

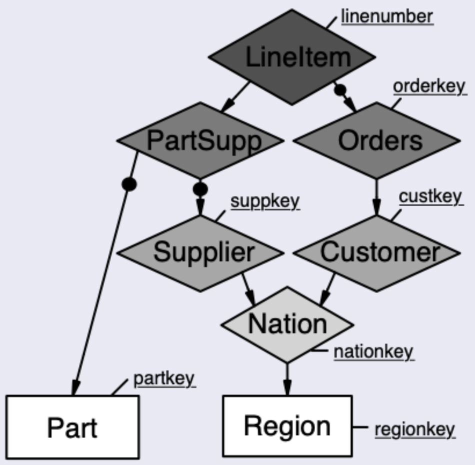


Relational semantics

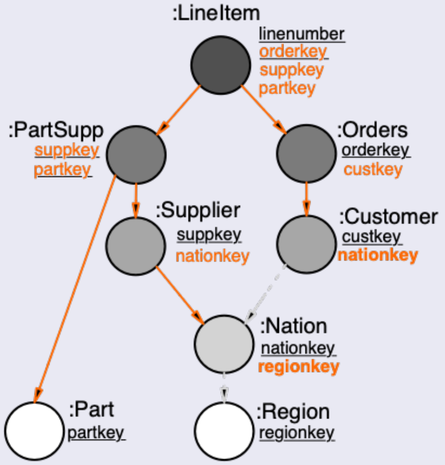


Relational semantics: duplicate key properties and use property keys

E/R graph model

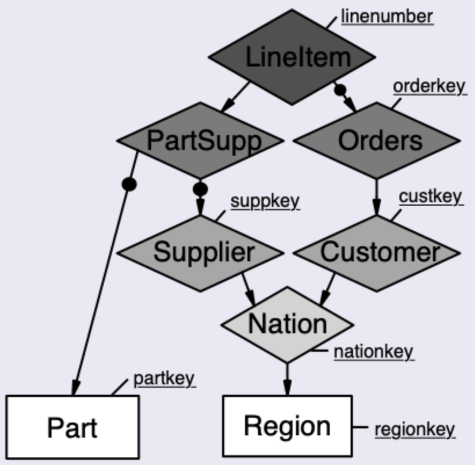


Relational semantics

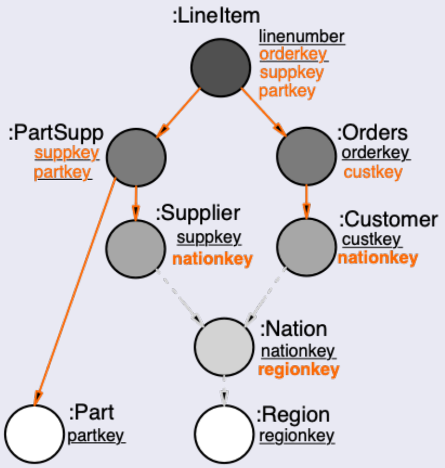


Relational semantics: duplicate key properties and use property keys

E/R graph model

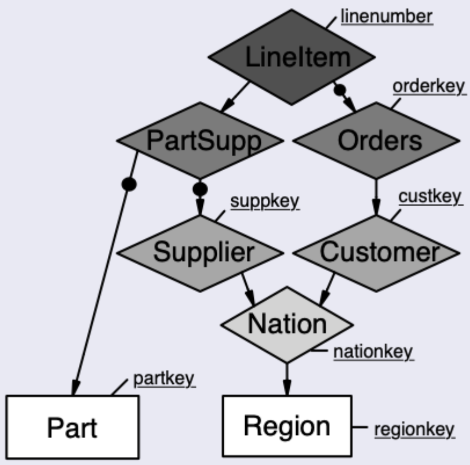


Relational semantics

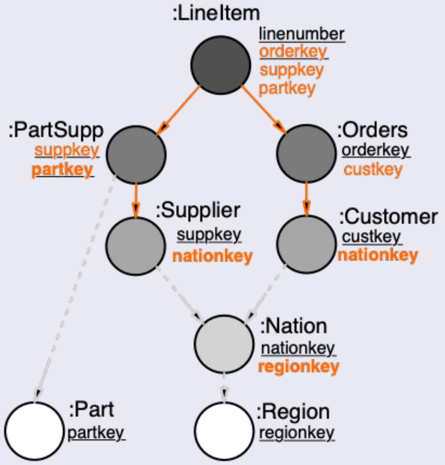


Relational semantics: duplicate key properties and use property keys

E/R graph model

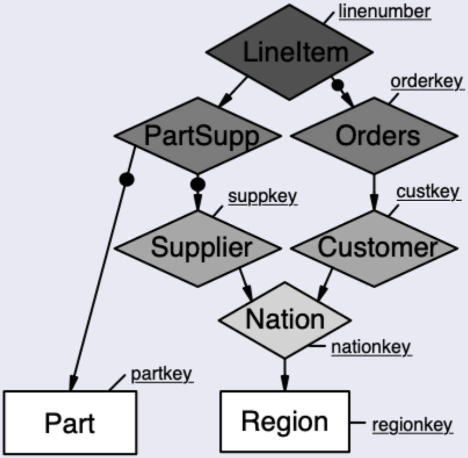


Relational semantics

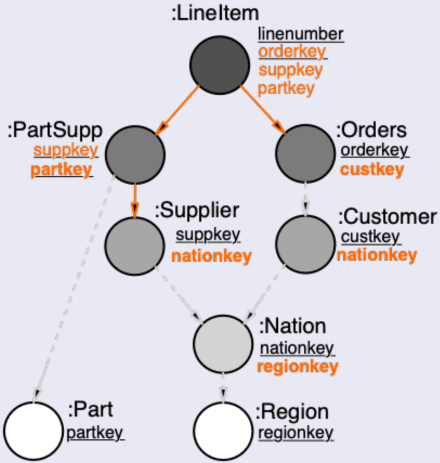


Relational semantics: duplicate key properties and use property keys

E/R graph model

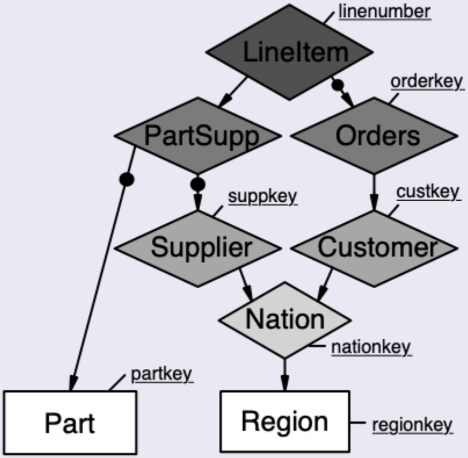


Relational semantics

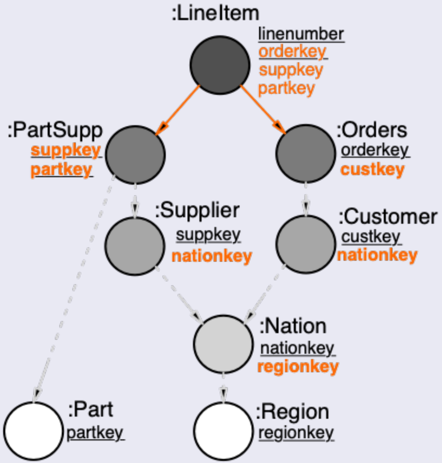


Relational semantics: duplicate key properties and use property keys

E/R graph model

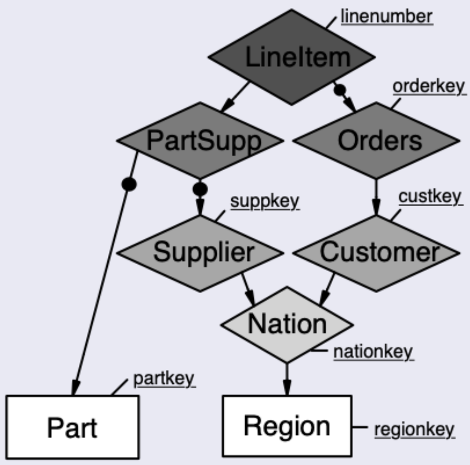


Relational semantics

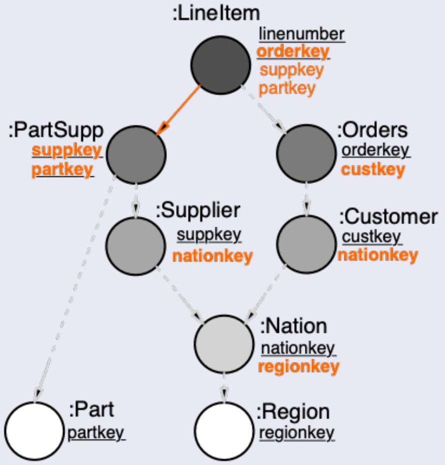


Relational semantics: duplicate key properties and use property keys

E/R graph model

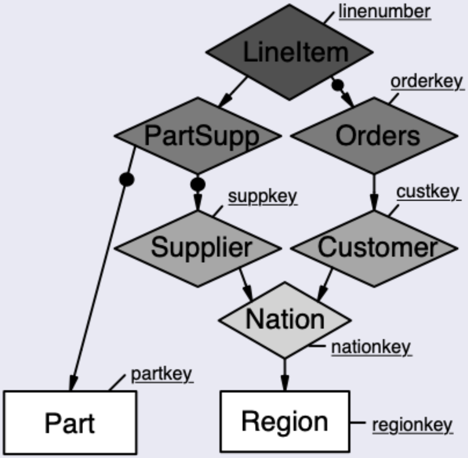


Relational semantics

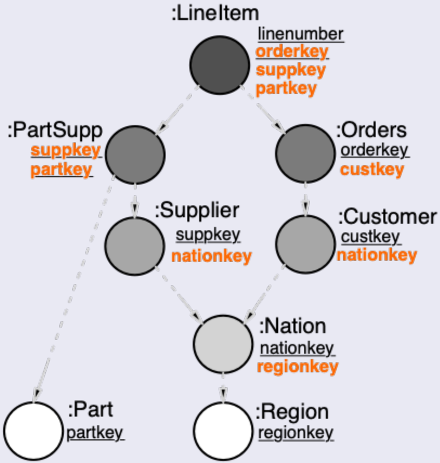


Relational semantics: duplicate key properties and use property keys

E/R graph model

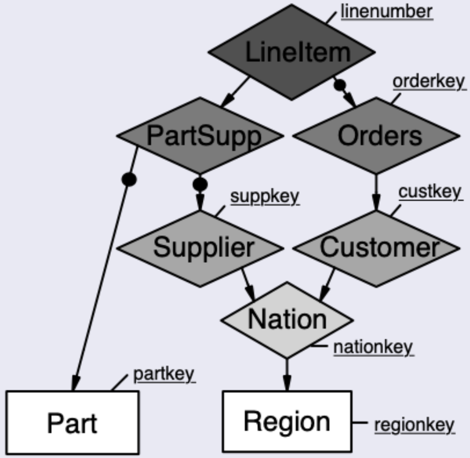


Relational semantics

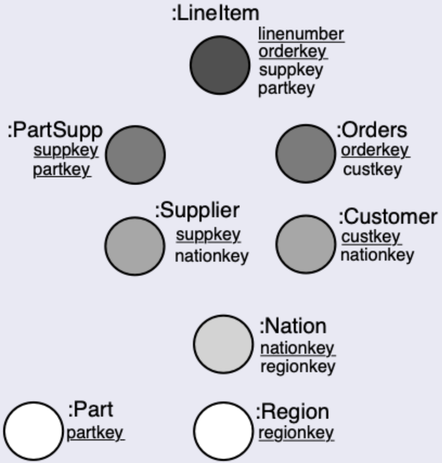


Relational semantics: duplicate key properties and use property keys

E/R graph model

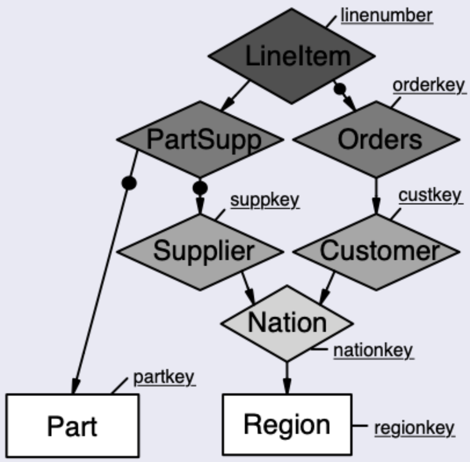


Relational semantics

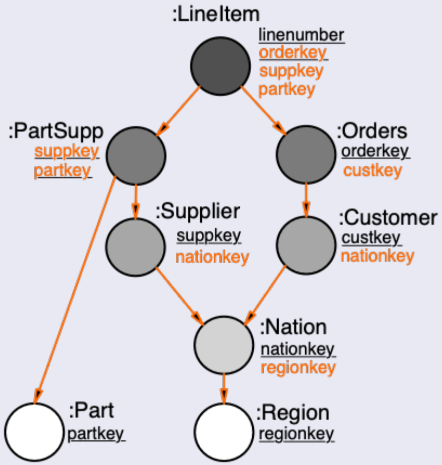


Graph semantics: no duplication of properties but need E/R keys

E/R graph model

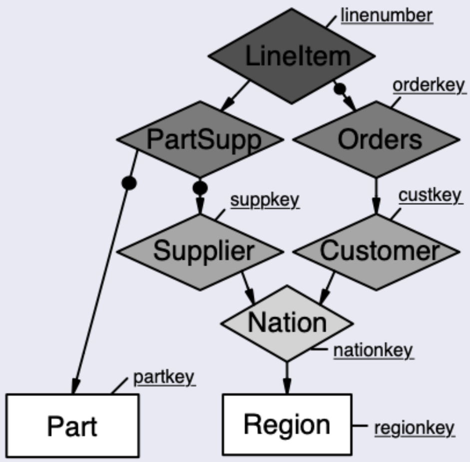


Graph semantics

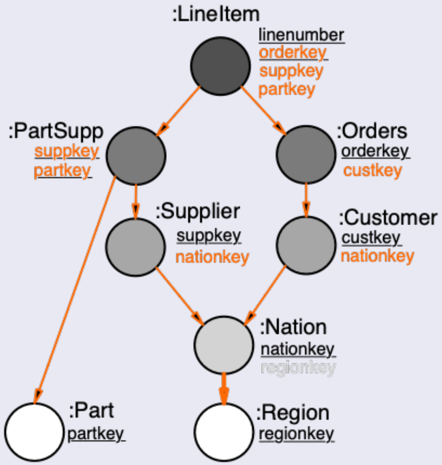


Graph semantics: no duplication of properties but need E/R keys

E/R graph model

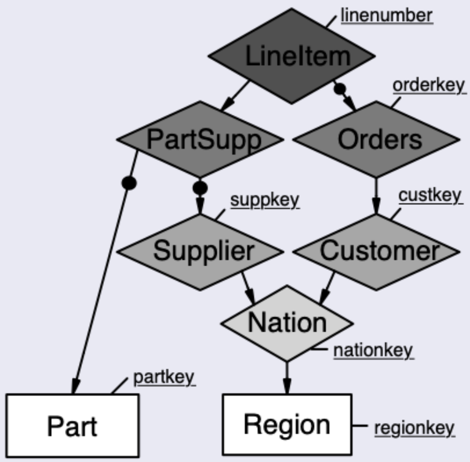


Graph semantics

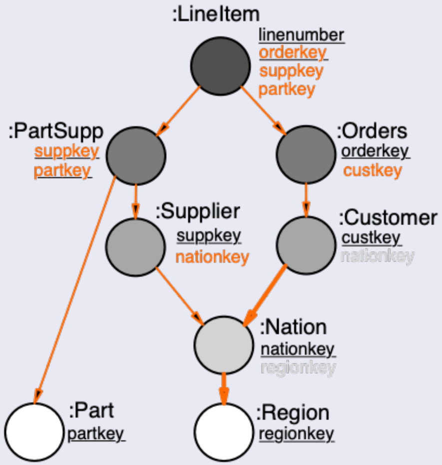


Graph semantics: no duplication of properties but need E/R keys

E/R graph model

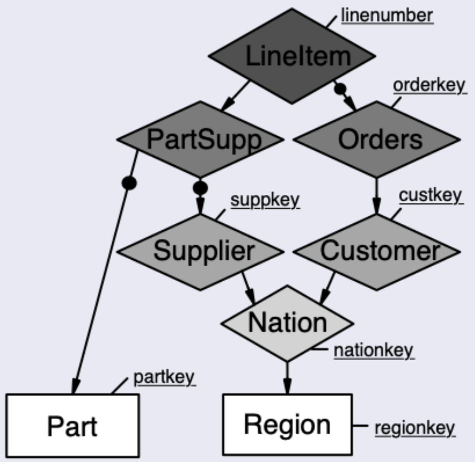


Graph semantics

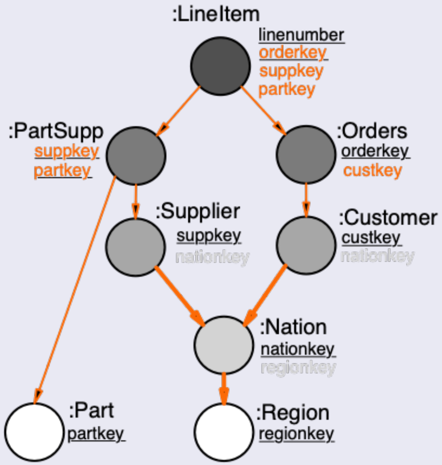


Graph semantics: no duplication of properties but need E/R keys

E/R graph model

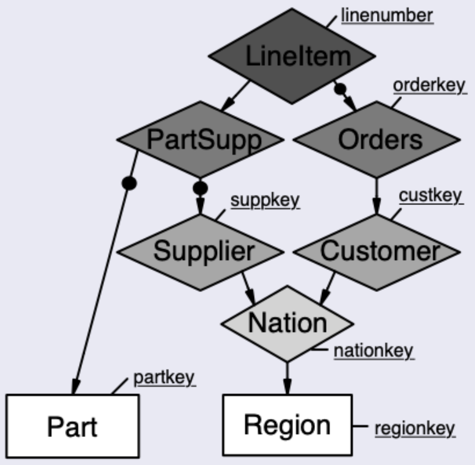


Graph semantics

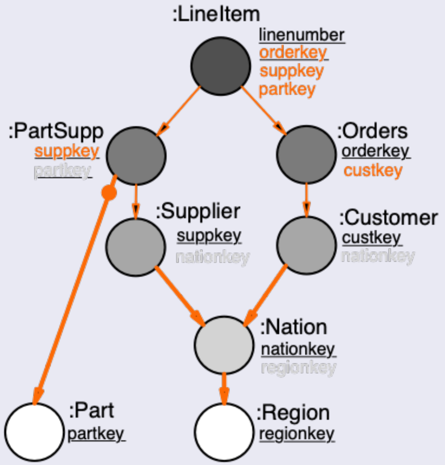


Graph semantics: no duplication of properties but need E/R keys

E/R graph model

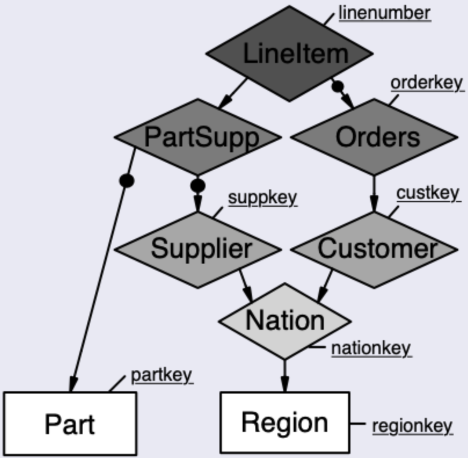


Graph semantics

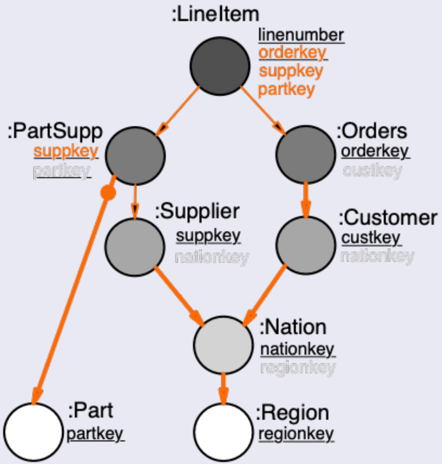


Graph semantics: no duplication of properties but need E/R keys

E/R graph model

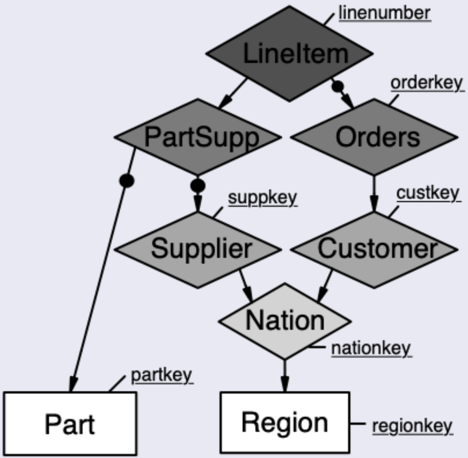


Graph semantics

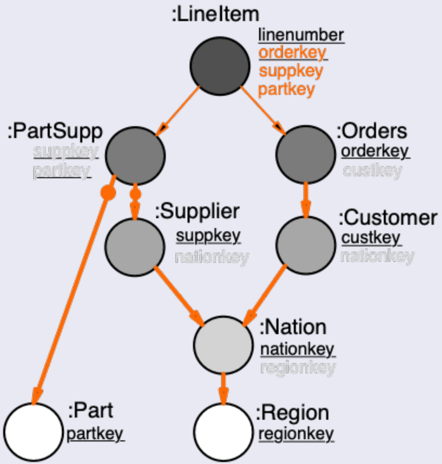


Graph semantics: no duplication of properties but need E/R keys

E/R graph model

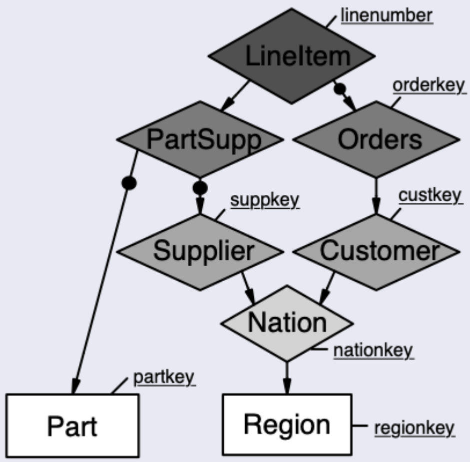


Graph semantics

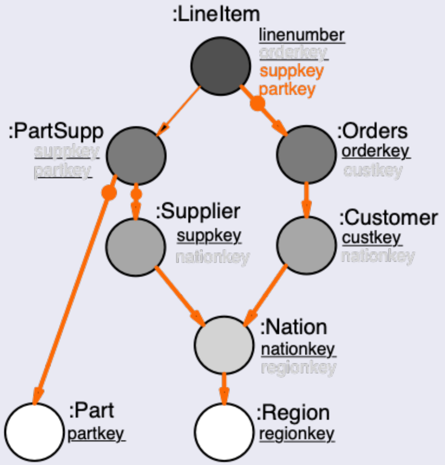


Graph semantics: no duplication of properties but need E/R keys

E/R graph model

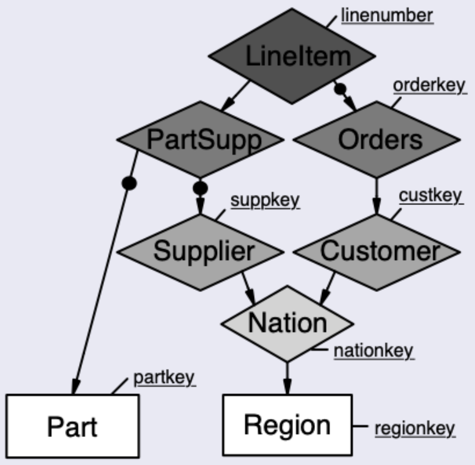


Graph semantics

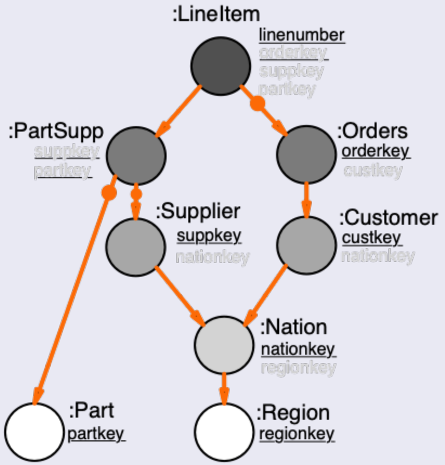


Graph semantics: no duplication of properties but need E/R keys

E/R graph model

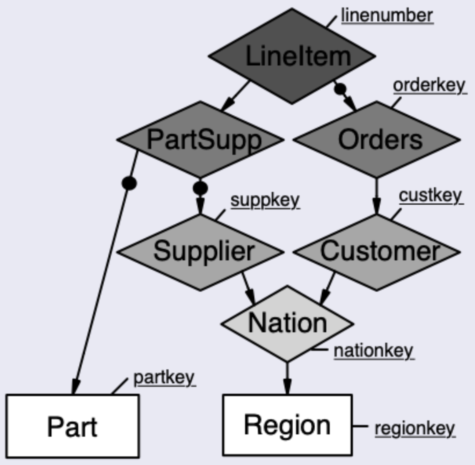


Graph semantics

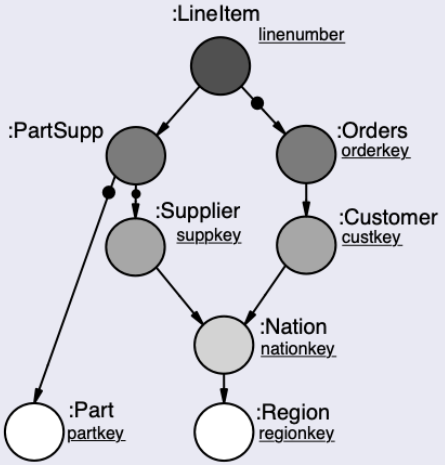


Graph semantics: no duplication of properties but need E/R keys

E/R graph model

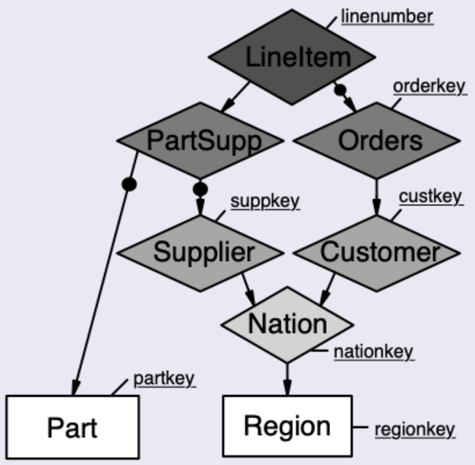


Graph semantics

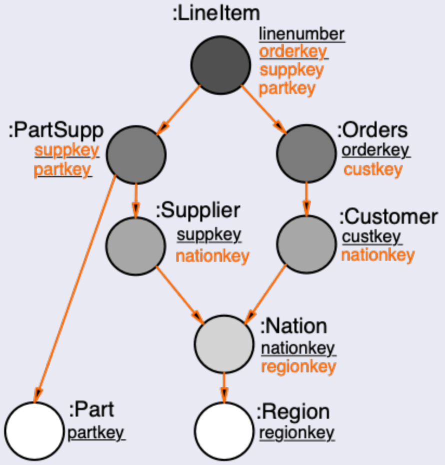


Mixed semantics: only keep E/R links to non-key components

E/R graph model

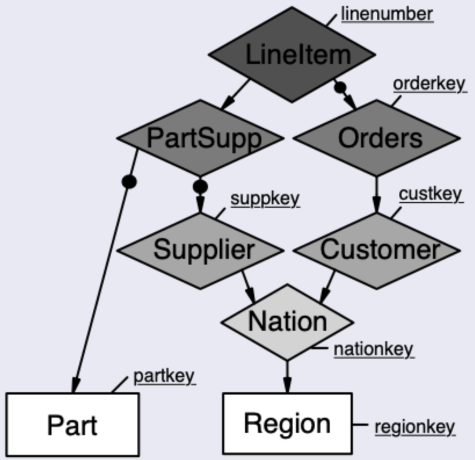


Mixed semantics

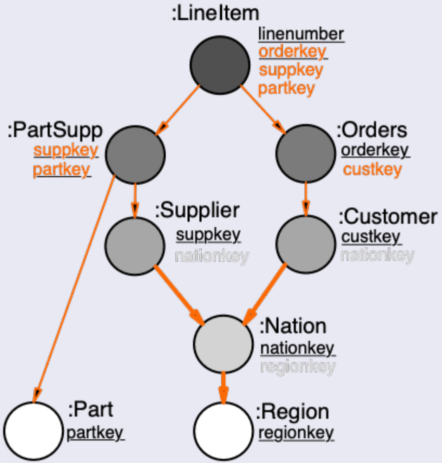


Mixed semantics: only keep E/R links to non-key components

E/R graph model

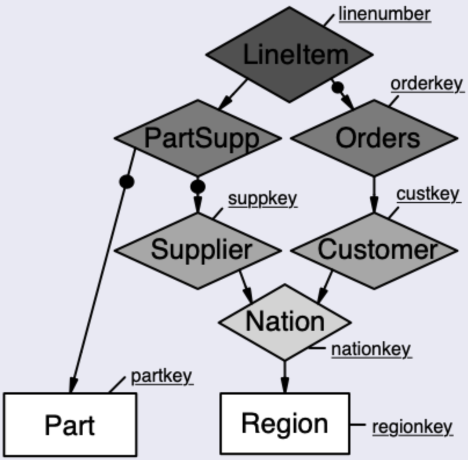


Mixed semantics

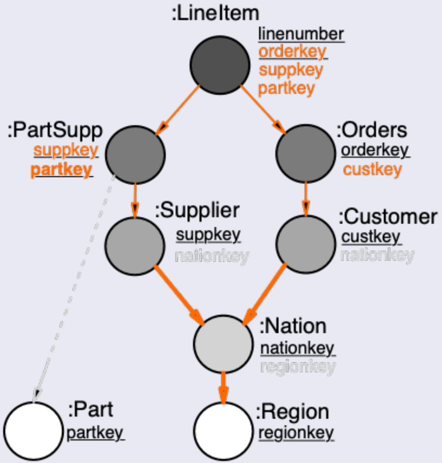


Mixed semantics: only keep E/R links to non-key components

E/R graph model

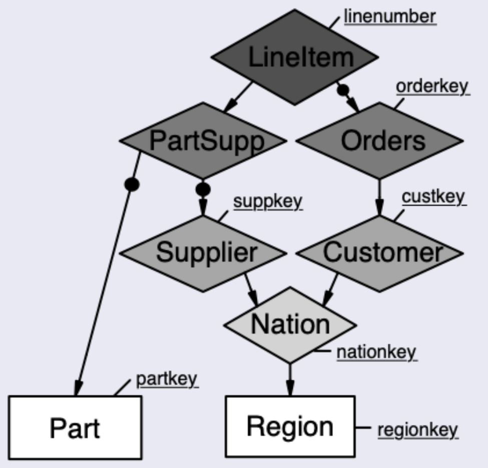


Mixed semantics

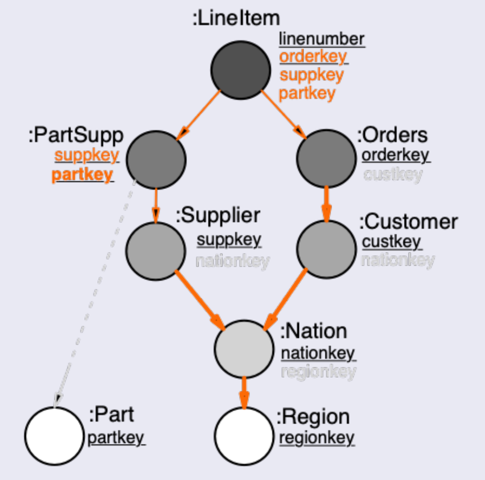


Mixed semantics: only keep E/R links to non-key components

E/R graph model

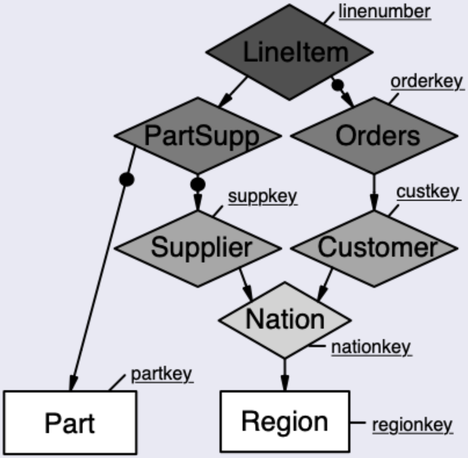


Mixed semantics

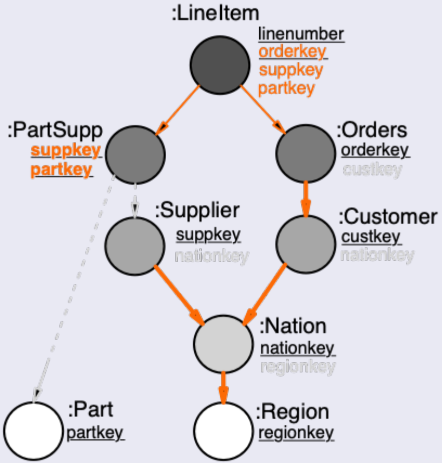


Mixed semantics: only keep E/R links to non-key components

E/R graph model

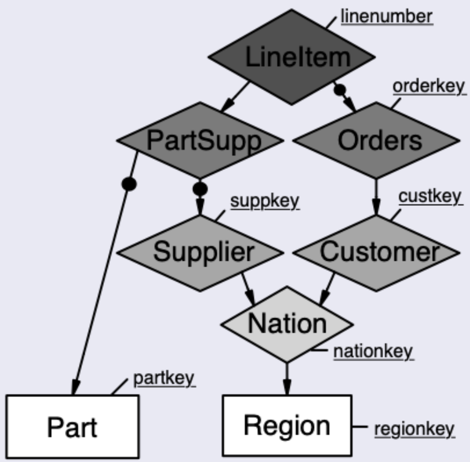


Mixed semantics

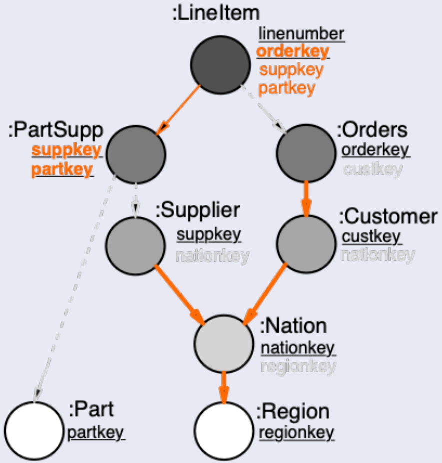


Mixed semantics: only keep E/R links to non-key components

E/R graph model

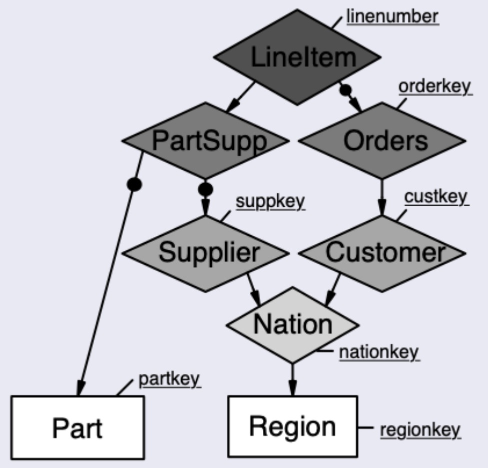


Mixed semantics

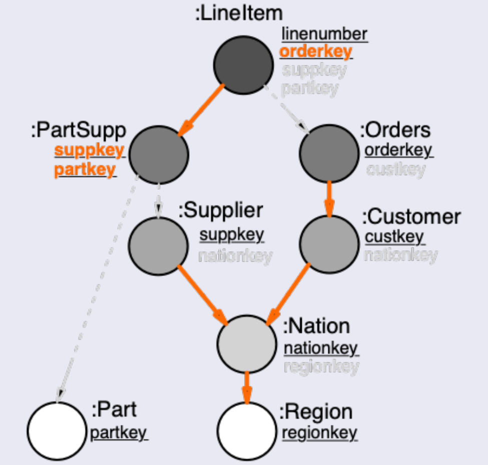


Mixed semantics: only keep E/R links to non-key components

E/R graph model

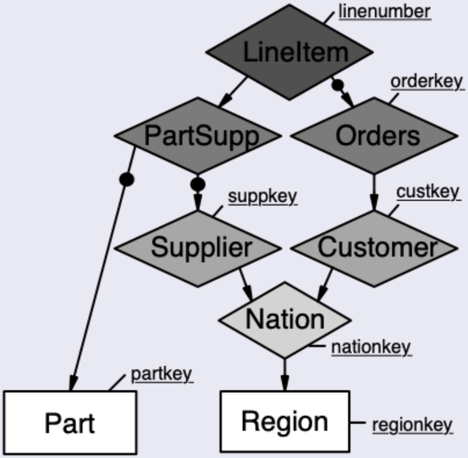


Mixed semantics

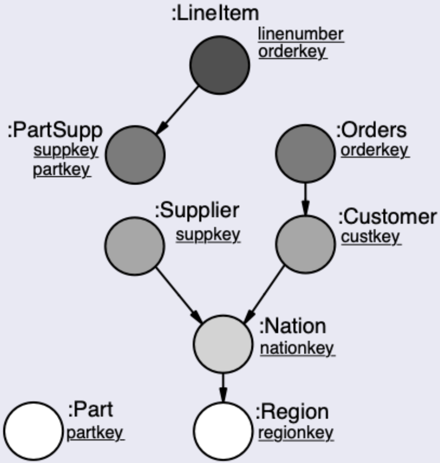


Mixed semantics: only keep E/R links to non-key components

E/R graph model



Mixed semantics



Formal Translations from Well-designed Relational Databases

Input

- Relational database schema $(\mathcal{S}, \Sigma_\mathcal{S})$ that is in Inclusion Dependency Normal Form (acyclic, key-based, foreign key-based)
- Relational database instance $\mathcal{I}(\mathcal{S})$ over $\mathcal{S}$ that satisfies $\Sigma_\mathcal{S}$

Order of relation schema $R \in \mathcal{S}$

The length of a longest key/foreign key chain from R to another relation schema $S \in \mathcal{S}$ on which no foreign key is defined

Relational Semantics

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R, and
- for every attribute $A \in R$ with $dom(A)$ we have a property-value pair $P_A = dom(A)$ on v_R,

Relational Semantics

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R, and
- for every attribute $A \in R$ with $dom(A)$ we have a property-value pair $P_A = dom(A)$ on v_R,
- for every key $\{A_1, \ldots, A_n\}$ of R we have a property key $R(\emptyset, \{P_{A_1}, \ldots, P_{A_n}\})$ implemented as PG-key:

  For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_n}$

Relational Semantics

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R, and
- for every attribute $A \in R$ with $dom(A)$ we have a property-value pair $P_A = dom(A)$ on v_R,
- for every key $\{A_1, \dots, A_n\}$ of R we have a property key $R(\emptyset, \{P_{A_1}, \dots, P_{A_n}\})$ implemented as PG-key:

  For (x:R) IDENTIFIER $x.P_{A_1}, \dots, x.P_{A_n}$
- for every foreign key $R[A_1, \dots, A_m] \subseteq S[B_1, \dots, B_m]$ of R we have a PG-key:

  For (x:R) IDENTIFIER $x.P_{A_1}, \dots, x.P_{A_m}, y$ WITHIN (x),(y:S) WHERE $x.P_{A_1} = y.P_{B_1}, \dots, x.P_{A_m} = y.P_{B_m}$.

Relational Semantics

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R, and
- for every attribute $A \in R$ with $dom(A)$ we have a property-value pair $P_A = dom(A)$ on v_R,
- for every key $\{A_1, \dots, A_n\}$ of R we have a property key $R(\emptyset, \{P_{A_1}, \dots, P_{A_n}\})$ implemented as PG-key:

  For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \dots, x.P_{A_n}$
- for every foreign key $R[A_1, \dots, A_m] \subseteq S[B_1, \dots, B_m]$ of R we have a PG-key:

  For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \dots, x.P_{A_m}, y$ WITHIN (x),$(y{:}S)$ WHERE $x.P_{A_1} = y.P_{B_1}, \dots, x.P_{A_m} = y.P_{B_m}$.

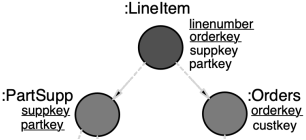
Secondly, the E/R graph obtained from an instance $\mathcal{I}(\mathcal{S})$ results from mapping every

- tuple $t \in \mathcal{I}(R)$ for every $R \in \mathcal{S}$ to a node v_t with label R such that, for all $A \in R$, $\nu(v_t, P_A) := t(A)$.

Example: Relational Semantics

Relation schema LINEITEM

- attributes *suppkey*, *partkey*, *orderkey*, *linenumber*, ..., *comment*
- key $\{linenumber, orderkey\}$ and foreign keys
  - $[suppkey, partkey] \subseteq \textsc{PartSupp}[suppkey, partkey]$ and
  - $[orderkey] \subseteq \textsc{Orders}[orderkey]$



Vertex $v_{\textsc{LineItem}}$ with label LINEITEM

- properties $P_{suppkey}$, $P_{partkey}$, $P_{orderkey}$, $P_{linenumber}$ ..., $P_{comment}$,
- E/R-key $\textsc{LineItem}(\emptyset, \{linenumber, orderkey\})$ implemented as PG-Key:

$$\text{For } (x{:}\textsc{LineItem}) \text{ IDENTIFIER } x.P_{linenumber}, x.P_{orderkey}$$

- Further PG-keys:
  - For $(x{:}\textsc{LineItem})$ IDENTIFIER $x.P_{suppkey}, x.P_{partkey}, y$ WITHIN $(x), (y{:}\textsc{PartSupp})$ WHERE $x.P_{suppkey} = y.P_{suppkey}, x.P_{partkey} = y.P_{partkey}$
  - For $(x{:}\textsc{LineItem})$ IDENTIFIER $x.P_{linenumber}, x.P_{orderkey}, y$ WITHIN $(x), (y{:}\textsc{Orders})$ WHERE $x.P_{orderkey} = y.P_{orderkey}$.

Graph Semantics

- R_{g}: Attributes of R that do not occur in any foreign key of R (others to be covered by E/R links)
- $C_R = \{S \in \mathcal{S} \mid \exists R[X] \subseteq S[Y] \in \Sigma_{\mathcal{S}}\}$ (the components of relationship type R)

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R, and
- for all $A : dom(A) \in R_{\mathrm{g}}$, we have a property-value pair $P_A = dom(A)$ of v_R,

Graph Semantics

- R_{g}: Attributes of R that do not occur in any foreign key of R (others to be covered by E/R links)
- $C_R = \{S \in \mathcal{S} \mid \exists R[X] \subseteq S[Y] \in \Sigma_{\mathcal{S}}\}$ (the components of relationship type R)

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R, and
- for all $A : dom(A) \in R_{\mathrm{g}}$, we have a property-value pair $P_A = dom(A)$ of v_R,
- for distinguished key K of R in Σ, we have an E/R key $R(C_K, P_K)$ where $C_K := \{S \in C_R \mid K \cap S \neq \emptyset\} = \{S_1, \dots, S_m\}$ and $P_K := \{P_A \mid A \in K\} - \{P_A \mid A \in S, S \in C_K\} = \{P_{A_1}, \dots, P_{A_n}\}$, implemented as:
  - directed edges $(v_R, v_{S_1}), \dots, (v_R, v_{S_m})$ with label $\bullet$ each,
  - For $(x{:}R)$ ID $x.P_{A_1}, \dots, x.P_{A_n}, y_1, \dots, y_m$ WITHIN $(x) \rightarrow (y_1{:}S_1), \dots, (x) \rightarrow (y_m{:}S_m)$

Graph Semantics

- R_{g}: Attributes of R that do not occur in any foreign key of R (others to be covered by E/R links)
- $C_R = \{S \in \mathcal{S} \mid \exists R[X] \subseteq S[Y] \in \Sigma_{\mathcal{S}}\}$ (the components of relationship type R)

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R, and
- for all $A : dom(A) \in R_{\mathrm{g}}$, we have a property-value pair $P_A = dom(A)$ of v_R,
- for distinguished key K of R in Σ, we have an E/R key $R(C_K, P_K)$ where $C_K := \{S \in C_R \mid K \cap S \neq \emptyset\} = \{S_1, \ldots, S_m\}$ and $P_K := \{P_A \mid A \in K\} - \{P_A \mid A \in S, S \in C_K\} = \{P_{A_1}, \ldots, P_{A_n}\}$, implemented as:
  - directed edges $(v_R, v_{S_1}), \ldots, (v_R, v_{S_m})$ with label $\bullet$ each,
  - For $(x{:}R)$ ID $x.P_{A_1}, \ldots, x.P_{A_n}, y_1, \ldots, y_m$ WITHIN $(x) {\rightarrow} (y_1{:}S_1), \ldots, (x) {\rightarrow} (y_m{:}S_m)$
- for every foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ of R where $S \notin C_K$, we have
  - a directed edge (v_R, v_S), and
  - a PG-key: For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_m}, y$ WITHIN $(x) \rightarrow (y{:}S)$.

Graph Semantics

- R_{g}: Attributes of R that do not occur in any foreign key of R (others to be covered by E/R links)
- $C_R = \{S \in \mathcal{S} \mid \exists R[X] \subseteq S[Y] \in \Sigma_{\mathcal{S}}\}$ (the components of relationship type R)

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R, and
- for all $A : dom(A) \in R_{\mathrm{g}}$, we have a property-value pair $P_A = dom(A)$ of v_R,
- for distinguished key K of R in Σ, we have an E/R key $R(C_K, P_K)$ where
  $C_K := \{S \in C_R \mid K \cap S \neq \emptyset\} = \{S_1, \dots, S_m\}$ and
  $P_K := \{P_A \mid A \in K\} - \{P_A \mid A \in S, S \in C_K\} = \{P_{A_1}, \dots, P_{A_n}\}$, implemented as:
    - directed edges $(v_R, v_{S_1}), \dots, (v_R, v_{S_m})$ with label $\bullet$ each,
    - For $(x{:}R)$ ID $x.P_{A_1}, \dots, x.P_{A_n}, y_1, \dots, y_m$ WITHIN $(x) \rightarrow (y_1{:}S_1), \dots, (x) \rightarrow (y_m{:}S_m)$
- for every foreign key $R[A_1, \dots, A_m] \subseteq S[B_1, \dots, B_m]$ of R where $S \notin C_K$, we have
    - a directed edge (v_R, v_S), and
    - a PG-key: For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \dots, x.P_{A_m}, y$ WITHIN $(x) \rightarrow (y{:}S)$.

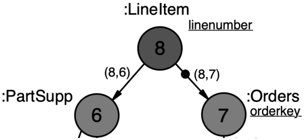
Secondly, the E/R graph obtained from an instance $\mathcal{I}(\mathcal{S})$ results from mapping for every $R \in \mathcal{S}$:

- $t \in \mathcal{I}(R) \mapsto$ node v_t with label R such that, $\forall A \in R_{\mathrm{g}}$, $\nu(v_t, P_A) := t(A)$
- for all $R[A_1, \dots, A_m] \subseteq S[B_1, \dots, B_m]$ of R, for all $t \in \mathcal{I}(R)$ with unique $s \in \mathcal{I}(S)$ such that $t[A_1, \dots, A_m] = s[B_1, \dots, B_m]$, we obtain a directed edge (v_t, v_s)

Example: Graph Semantics

Relation schema LINEITEM

- attributes *suppkey*, *partkey*, *orderkey*, *linenumber*, . . . , *comment*
- key $\{linenumber, orderkey\}$ and foreign keys
  - $[suppkey, partkey] \subseteq$ PARTSUPP$[suppkey, partkey]$ and
  - $[orderkey] \subseteq$ ORDERS$[orderkey]$



Vertex v_{LINEITEM} with label LINEITEM

- properties $P_{linenumber}, \ldots, P_{comment}$ ($P_{suppkey}$, $P_{partkey}$, $P_{orderkey}$ not required)
- an E/R-key LINEITEM($\{$ORDERS$\}$, $\{linenumber\}$) implemented as
  - For (x:LINEITEM) IDENTIFIER $x.P_{linenumber}, y$ WITHIN $(x) \rightarrow$ (y:ORDERS)
- further PG-key:
  - For (x:LINEITEM) IDENTIFIER x, y WITHIN $(x) \rightarrow$ (y:PARTSUPP).

Note that relation schema ORDERS has the key $\{orderkey\}$, and relation schema PARTSUPP has the key $\{partkey, suppkey\}$.

Mixed Semantics

- $R_{\text{mix}} := R - (F_R - K_R)$: remove attributes that belong to some foreign key but not to the key
- No foreign key for which some of its attributes belong to the key and some others do not

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R,
- for all $A : dom(A) \in R_{\text{mix}}$, we have a property-value pair $P_A = dom(A)$ of v_R,

Mixed Semantics

- $R_{\text{mix}} := R - (F_R - K_R)$: remove attributes that belong to some foreign key but not to the key
- No foreign key for which some of its attributes belong to the key and some others do not

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R,
- for all $A : dom(A) \in R_{\text{mix}}$, we have a property-value pair $P_A = dom(A)$ of v_R,
- for key $\{A_1, \ldots, A_n\}$ of R, we have property key $R(\emptyset, \{P_{A_1}, \ldots, P_{A_n}\})$ implemented as PG-key: For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_n}$

Mixed Semantics

- $R_{\text{mix}} := R - (F_R - K_R)$: remove attributes that belong to some foreign key but not to the key
- No foreign key for which some of its attributes belong to the key and some others do not

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R,
- for all $A : dom(A) \in R_{\text{mix}}$, we have a property-value pair $P_A = dom(A)$ of v_R,
- for key $\{A_1, \ldots, A_n\}$ of R, we have property key $R(\emptyset, \{P_{A_1}, \ldots, P_{A_n}\})$ implemented as PG-key: For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_n}$
- for foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ where $\{A_1, \ldots, A_m\} \subseteq R_{\text{mix}}$, we have PG-key:
  - For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_m}, y$ WITHIN (x),$(y{:}S)$ WHERE $x.P_{A_1} = y.P_{B_1}, \ldots, x.P_{A_m} = y.P_{B_m}$

Mixed Semantics

- $R_{\text{mix}} := R - (F_R - K_R)$: remove attributes that belong to some foreign key but not to the key
- No foreign key for which some of its attributes belong to the key and some others do not

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R,
- for all $A : dom(A) \in R_{\text{mix}}$, we have a property-value pair $P_A = dom(A)$ of v_R,
- for key $\{A_1, \ldots, A_n\}$ of R, we have property key $R(\emptyset, \{P_{A_1}, \ldots, P_{A_n}\})$ implemented as PG-key: For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_n}$
- for foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ where $\{A_1, \ldots, A_m\} \subseteq R_{\text{mix}}$, we have PG-key:
  - For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_m}, y$ WITHIN (x),$(y{:}S)$ WHERE $x.P_{A_1} = y.P_{B_1}, \ldots, x.P_{A_m} = y.P_{B_m}$
- for foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ of R where $\{A_1, \ldots, A_m\} \cap R_{\text{mix}} = \emptyset$, we have
  - a directed edge (v_R, v_S)
  - a PG-key: For $(x{:}R)$ IDENTIFIER x, y WITHIN $(x) \rightarrow (y{:}S)$.

Mixed Semantics

- $R_{\text{mix}} := R - (F_R - K_R)$: remove attributes that belong to some foreign key but not to the key
- No foreign key for which some of its attributes belong to the key and some others do not

Firstly, the E/R graph model is obtained by mapping every relation schema $R \in \mathcal{S}$ to

- a vertex v_R with label R,
- for all $A : dom(A) \in R_{\text{mix}}$, we have a property-value pair $P_A = dom(A)$ of v_R,
- for key $\{A_1, \ldots, A_n\}$ of R, we have property key $R(\emptyset, \{P_{A_1}, \ldots, P_{A_n}\})$ implemented as PG-key: For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_n}$
- for foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ where $\{A_1, \ldots, A_m\} \subseteq R_{\text{mix}}$, we have PG-key:
  - For $(x{:}R)$ IDENTIFIER $x.P_{A_1}, \ldots, x.P_{A_m}, y$ WITHIN (x),$(y{:}S)$ WHERE $x.P_{A_1} = y.P_{B_1}, \ldots, x.P_{A_m} = y.P_{B_m}$
- for foreign key $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ of R where $\{A_1, \ldots, A_m\} \cap R_{\text{mix}} = \emptyset$, we have
  - a directed edge (v_R, v_S)
  - a PG-key: For $(x{:}R)$ IDENTIFIER x, y WITHIN $(x) \rightarrow (y{:}S)$.

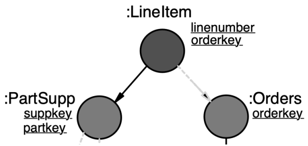
Secondly, the E/R graph obtained from an instance $\mathcal{I}(\mathcal{S})$ results from mapping for every $R \in \mathcal{S}$:

- $t \in \mathcal{I}(R) \mapsto$ node v_t with label R such that, $\forall A \in R_{\text{mix}}$, $\nu(v_t, P_A) := t(A)$
- for all $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$ where $\{A_1, \ldots, A_m\} \cap R_{\text{mix}} = \emptyset$, for all $t \in \mathcal{I}(R)$ with unique $s \in \mathcal{I}(S)$ such that $t[A_1, \ldots, A_m] = s[B_1, \ldots, B_m]$, we obtain a directed edge (v_t, v_s)

Example: Mixed Semantics

Relation schema $\textsc{LineItem}$

- attributes *suppkey*, *partkey*, *orderkey*, *linenumber*, ..., *comment*
- key $\{linenumber, orderkey\}$ and foreign keys
  - $[suppkey, partkey] \subseteq \textsc{PartSupp}[suppkey, partkey]$ and
  - $[orderkey] \subseteq \textsc{Orders}[orderkey]$



Vertex $v_{\textsc{LineItem}}$ with label $\textsc{LineItem}$

- $P_{orderkey}, P_{linenumber}, \ldots, P_{comment}$ ($P_{suppkey}$ and $P_{partkey}$ are not required),
- a property key $\textsc{LineItem}(\emptyset, \{linenumber, orderkey\})$ implemented as
  - For $(x{:}\textsc{LineItem})$ IDENTIFIER $x.P_{linenumber}, x.P_{orderkey}$
- and further PG-keys:
  - For $(x{:}\textsc{LineItem})$ IDENTIFIER x, y WITHIN $(x) \rightarrow (y{:}\textsc{PartSupp})$, and
  - For $(x{:}\textsc{LineItem})$ IDENTIFIER $x.P_{linenumber}, x.P_{orderkey}, y$ WITHIN $(x), (y{:}\textsc{Orders})$ WHERE $x.P_{orderkey} = y.P_{orderkey}$.

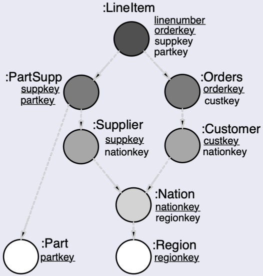
Note that relation schema $\textsc{PartSupp}$ has the key $\{partkey, suppkey\}$.

Pitstop 4

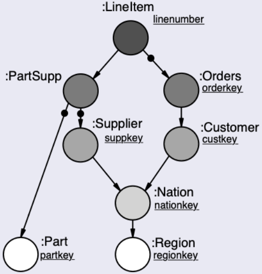
Three Principles Choices for Managing Entity and Referential (E/R) Integrity

- Relational: Use property keys uniformly by admitting property redundancy
- Graphs: Use E/R keys uniformly to eliminate property redundancy
- Mixed: Rely on property keys only but minimize property redundancy

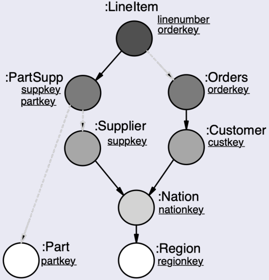
Relational semantics



Graph semantics



Mixed semantics



Experiments

Summary

Question

How well is data integrity managed under relational, mixed, and graph semantics?

Summary

Question

How well is data integrity managed under relational, mixed, and graph semantics?

Approach

Translate TPC-H database into Neo4j using different scaling factors and semantics, and then measure the effort of maintaining entity and referential integrity.

Summary

Question

How well is data integrity managed under relational, mixed, and graph semantics?

Approach

Translate TPC-H database into Neo4j using different scaling factors and semantics, and then measure the effort of maintaining entity and referential integrity.

Main findings

- Graph semantics can have savings over the other semantics in orders of magnitude
- E/R keys do not enjoy native support in graph database systems yet
- Mixed semantics most efficient in current limits that only support property keys

Summary

Question

How well is data integrity managed under relational, mixed, and graph semantics?

Approach

Translate TPC-H database into Neo4j using different scaling factors and semantics, and then measure the effort of maintaining entity and referential integrity.

Main findings

- Graph semantics can have savings over the other semantics in orders of magnitude
- E/R keys do not enjoy native support in graph database systems yet
- Mixed semantics most efficient in current limits that only support property keys

- TPC-H in IDNF, diverse enough to quantify (dis-)advantages of the semantics
  1. Key/foreign key chains of different lengths
  2. E/R and property keys
  3. Different data volumes

Set Up

For each of the three semantics, we translated the TPC-H database into Neo4j using scaling factors small (0.01), medium (0.1) and large (1).

Semantics	sf=0.01		sf=0.1		sf=1	
	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$
Relational	86,805	0	866,602	0	8,661,245	0
Mixed	86,805	76,800	866,602	766,597	8,661,245	7,661,240
Graph	86,805	152,975	866,602	1,527,169	8,661,245	15,262,455

Set Up

For each of the three semantics, we translated the TPC-H database into Neo4j using scaling factors small (0.01), medium (0.1) and large (1).

Semantics	*sf*=0.01		*sf*=0.1		*sf*=1	
	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$
Relational	86,805	0	866,602	0	8,661,245	0
Mixed	86,805	76,800	866,602	766,597	8,661,245	7,661,240
Graph	86,805	152,975	866,602	1,527,169	8,661,245	15,262,455

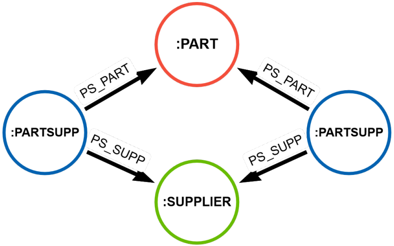
Specifications

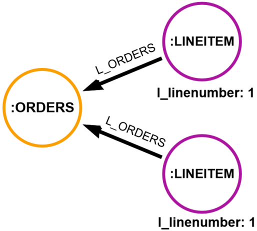
- Neo4j with its query language Cypher in conjunction with Python 3.9.13
- 64-bit operating system with an Intel Core i7 Processor and 16GB RAM

`https://github.com/graphdbexperiments/er_graph_experiments`

Entity Integrity

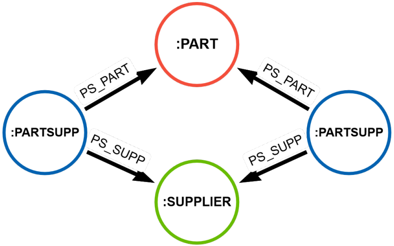
- Perform Cypher queries that validate the E/R keys
  - $\mathrm{PS}_k = \mathrm{PS}(\{\textsc{Part}, \textsc{Supp}\}, \emptyset)$ on $\textsc{PartSupp}$
  - $\mathrm{L}_k = \mathrm{L}(\{\textsc{Orders}\}, \{\mathit{linenumber}\})$ on $\textsc{LineItem}$

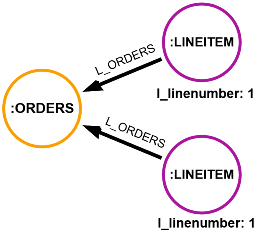




Entity Integrity

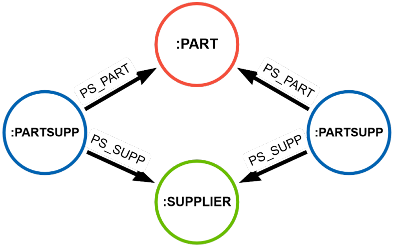
- Perform Cypher queries that validate the E/R keys
  - $\mathrm{PS}_k = \mathrm{PS}(\{\textsc{Part}, \textsc{Supp}\}, \emptyset)$ on $\textsc{PartSupp}$
  - $\mathrm{L}_k = \mathrm{L}(\{\textsc{Orders}\}, \{\mathit{linenumber}\})$ on $\textsc{Lineitem}$
- Under relational (r) and mixed (m) semantics, we can specify these keys as unique constraints within Neo4j, and benefit from the resulting index.

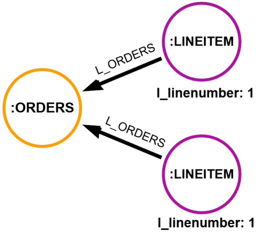




Entity Integrity

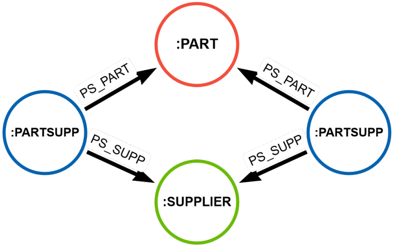
- Perform Cypher queries that validate the E/R keys
  - $\mathrm{PS}_k = \mathrm{PS}(\{\textsc{Part}, \textsc{Supp}\}, \emptyset)$ on $\textsc{PartSupp}$
  - $\mathrm{L}_k = \mathrm{L}(\{\textsc{Orders}\}, \{\textit{linenumber}\})$ on $\textsc{Lineitem}$
- Under relational (r) and mixed (m) semantics, we can specify these keys as unique constraints within Neo4j, and benefit from the resulting index.
- Under graph semantics (g), the E/R keys are not supported by Neo4j (or any other graph system), so we cannot benefit from any index structure either.

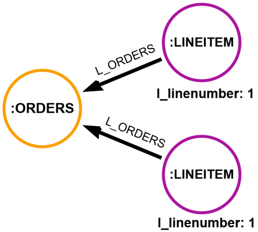




Entity Integrity

- Perform Cypher queries that validate the E/R keys
  - $\mathrm{PS}_k = \mathrm{PS}(\{\textsc{Part}, \textsc{Supp}\}, \emptyset)$ on $\textsc{PartSupp}$
  - $\mathrm{L}_k = \mathrm{L}(\{\textsc{Orders}\}, \{\mathit{linenumber}\})$ on $\textsc{Lineitem}$
- Under relational (r) and mixed (m) semantics, we can specify these keys as unique constraints within Neo4j, and benefit from the resulting index.
- Under graph semantics (g), the E/R keys are not supported by Neo4j (or any other graph system), so we cannot benefit from any index structure either.
- Under all semantics, the key is satisfied whenever a corresponding Cypher query returns no nodes violating the key

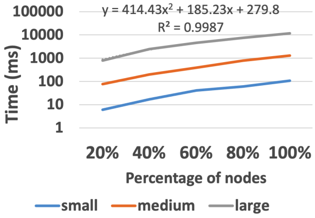




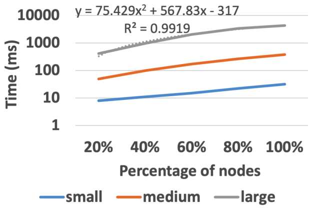
E/R keys used for validation experiments

Semantics	E/R key	Cypher Query	Index
Relational and Mixed	$\mathrm{PS}_k(r) = \mathrm{PS}_k(m)$: PS(∅,{partkey, suppkey})	MATCH (ps1:PS), (ps2:PS) WHERE id(ps1) < id(ps2) AND ps1.partkey = ps2.partkey AND ps1.suppkey = ps2.suppkey RETURN ps1, ps2	PS(partkey, suppkey)
Graph	$\mathrm{PS}_k(g)$: PS({P, S},∅)	MATCH q1 = (s:S)<-[]-(ps1:PS)-[]->(p:P), q2=(s:S)<-[]-(ps2:PS)-[]->(p:P) WHERE id(ps1) < id(ps2) RETURN q1, q2	no index
Relational and Mixed	$\mathrm{L}_k(r) = \mathrm{L}_k(m)$: L(∅,{orderkey, linenumber})	MATCH (l1:L), (l2:L) WHERE id(l1) < id(l2) AND l1.orderkey = l2.orderkey AND l1.linenumber = l2.linenumber RETURN l1, l2	L(orderkey, linenumber)
Graph	$\mathrm{L}_k(g)$: L({O},linenumber)	MATCH q = (l1:L)-[]->(o:O)<-[]-(l2:L) WHERE id(l1) < id(l2) AND l1.linenumber = l2.linenumber RETURN q	no index

How well does E/R Key validation scale?

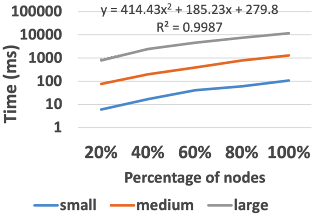


(a) PS_k: r and m semantics

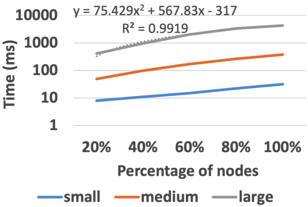


(b) PS_k: g semantics

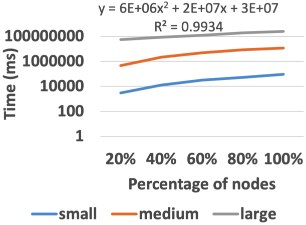
How well does E/R Key validation scale?



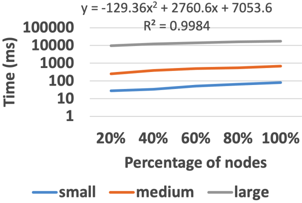
(a) PS_k: r and m semantics



(b) PS_k: g semantics

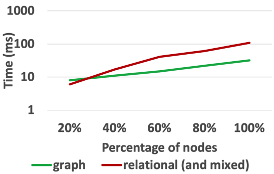


(c) L_k: r and m semantics

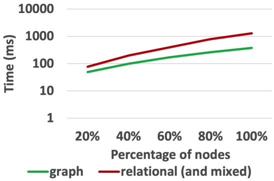


(d) L_k: g semantics

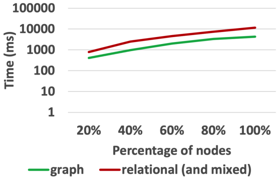
How does E/R Key validation compare across the different semantics?



(a) PartSupp: sf=0.01

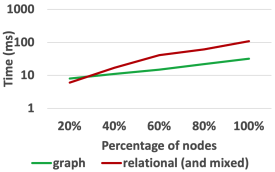


(b) PartSupp: sf=0.1

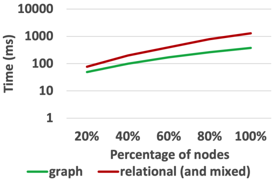


(c) PartSupp: sf=1

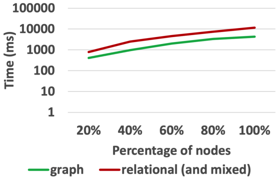
How does E/R Key validation compare across the different semantics?



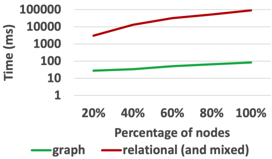
(a) PartSupp: sf=0.01



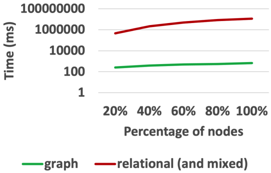
(b) PartSupp: sf=0.1



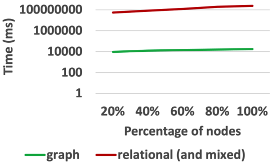
(c) PartSupp: sf=1



(d) Lineitem: sf=0.01



(e) Lineitem: sf=0.1



(f) Lineitem: sf=1

Measuring E/R key validation efficiency

E/R key	sem	*sf*=0.01 db hits	*sf*=0.01 time (ms)	*sf*=0.1 db hits	*sf*=0.1 time (ms)	*sf*=1 db hits	*sf*=1 time (ms)
PS_k	r/m	16,002	108	160,002	1,288	1,600,002	11,669
	g	442,701	32	4,302,889	377	23,282,432	4,310
L_k	r/m	120,352	91,258	1,201,146	11,562,701	26,400,286	246,985,775
	g	842,452	82	8,860,929	686	88,530,831	17,696

- DB hits estimate efficiency independent of size but in number of access operations
- Poor db hits for graph semantics is due to absence of any underlying index
- Actual validation time much more efficient than for relational and graph semantics
- Due to focus of graph database systems on efficient processing of edges
- Huge potential for further improvements once indices available for E/R keys

Referential Integrity

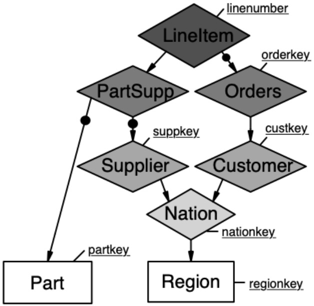
Key/foreign key chains

P_c: $\textsc{Part}(P) \subset \textsc{PartSupp}(PS) \subset \textsc{LineItem}(L)$

S_c: $\textsc{Supplier}(S) \subset \textsc{PartSupp}(PS) \subset \textsc{LineItem}(L)$

C_c: $\textsc{Customer}(C) \subset \textsc{Orders}(O)$

O_c: $\textsc{Orders}(O) \subset \textsc{LineItem}(L)$



Referential Integrity

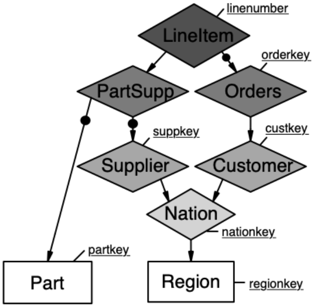
Key/foreign key chains

P_c: $\textsc{Part}(P) \subset \textsc{PartSupp}(PS) \subset \textsc{LineItem}(L)$

S_c: $\textsc{Supplier}(S) \subset \textsc{PartSupp}(PS) \subset \textsc{LineItem}(L)$

C_c: $\textsc{Customer}(C) \subset \textsc{Orders}(O)$

O_c: $\textsc{Orders}(O) \subset \textsc{LineItem}(L)$



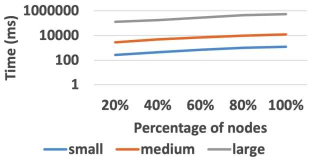
S_c : Update of *suppkey* value

- for different percentages of SUPPLIER-nodes (all semantics)
- requires updates for *suppkey* on PARTSUPP-nodes (relational and mixed)
- which requires updates for *suppkey* on LINEITEM-nodes (for relational only)

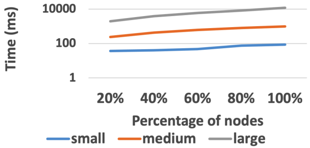
Update propagation for $S \subset PS \subset L$ by semantics

Semantics	Update propagation
Relational	MATCH (s:S), (ps:PS), (l:L) WHERE ps.suppkey = s.suppkey AND l.suppkey = s.suppkey SET s.suppkey = RIGHT(('00000000' + toString(s.suppkey)), 8), ps.suppkey = RIGHT(('00000000' + toString(ps.suppkey)), 8), l.suppkey = RIGHT(('00000000'+toString(l.suppkey)), 8)
Mixed	MATCH (s:S), (ps:PS) WHERE ps.suppkey = s.suppkey SET s.suppkey = RIGHT(('00000000' + toString(s.suppkey)), 8), ps.suppkey = RIGHT(('00000000' + toString(ps.suppkey)), 8)
Graph	MATCH (s:S) SET s.suppkey = RIGHT(('00000000' + toString(s.suppkey)), 8)

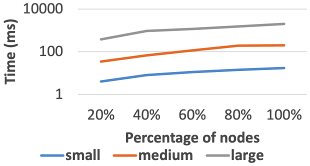
How well does update propagation scale? - Part 1/2



(a) Relational: $P \subset PS \subset L$

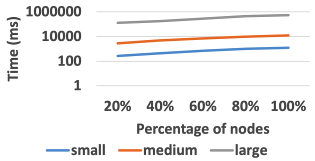


(b) Mixed: $P \subset PS \subset L$

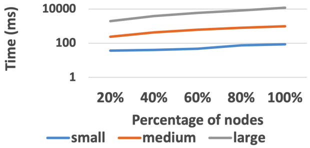


(c) Graph: $P \subset PS \subset L$

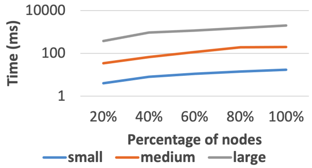
How well does update propagation scale? - Part 1/2



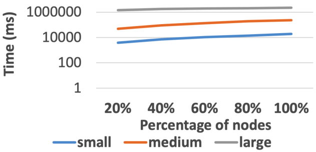
(a) Relational: $P \subset PS \subset L$



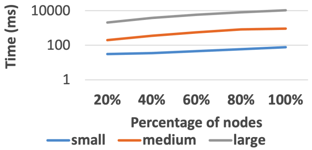
(b) Mixed: $P \subset PS \subset L$



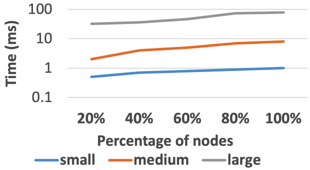
(c) Graph: $P \subset PS \subset L$



(d) Relational: $S \subset PS \subset L$

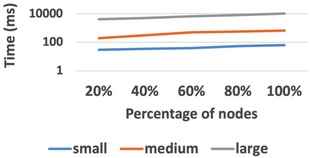


(e) Mixed: $S \subset PS \subset L$

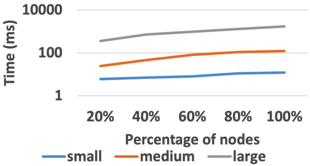


(f) Graph: $S \subset PS \subset L$

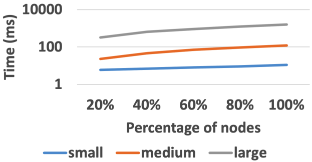
How well does update propagation scale? - Part 2/2



(a) Relational: $C \subset O$

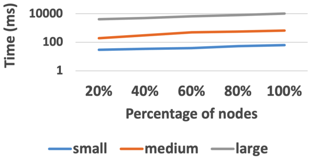


(b) Mixed: $C \subset O$

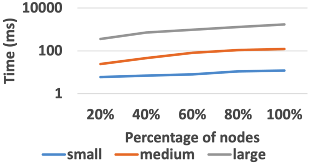


(c) Graph: $C \subset O$

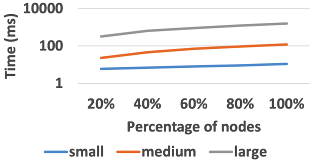
How well does update propagation scale? - Part 2/2



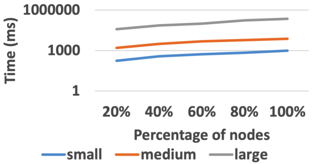
(a) Relational: $C \subset O$



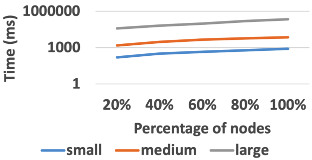
(b) Mixed: $C \subset O$



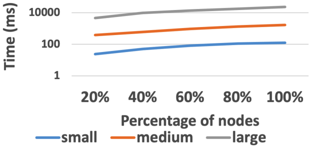
(c) Graph: $C \subset O$



(d) Relational: $O \subset L$

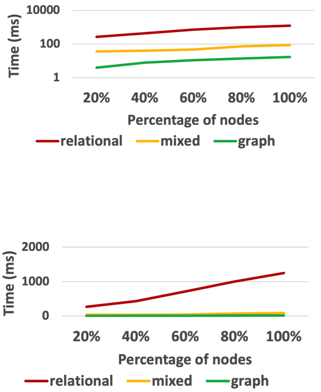


(e) Mixed: $O \subset L$

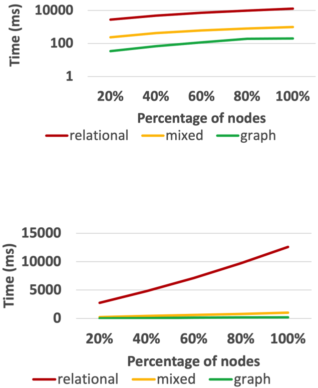


(f) Graph: $O \subset L$

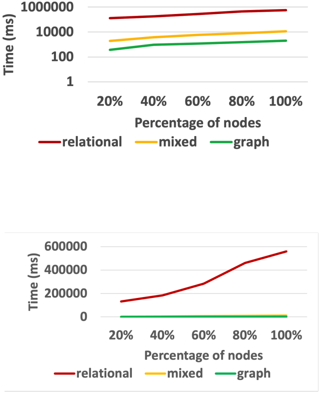
How well does update propagation work under different semantics? - 1/4



(a) *sf=0.01*: $P \subset PS \subset L$

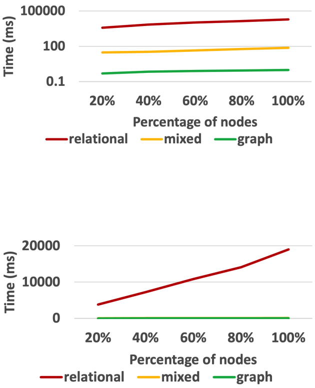


(b) *sf=0.1*: $P \subset PS \subset L$

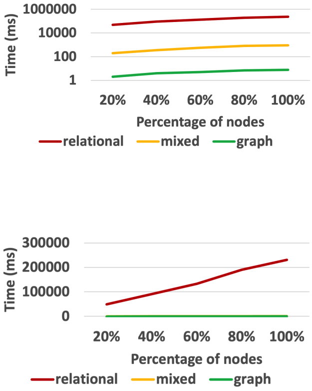


(c) *sf=1*: $P \subset PS \subset L$

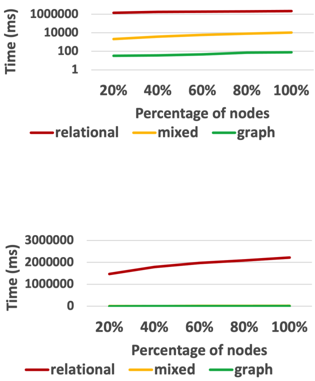
How well does update propagation work under different semantics? - 2/4



(a) *sf*=0.01: $S \subset PS \subset L$

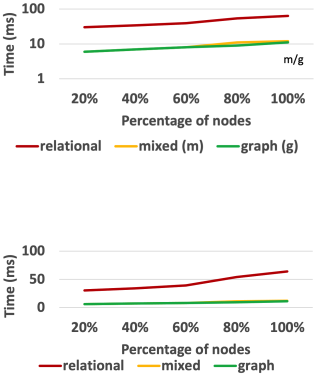


(b) *sf*=0.1: $S \subset PS \subset L$

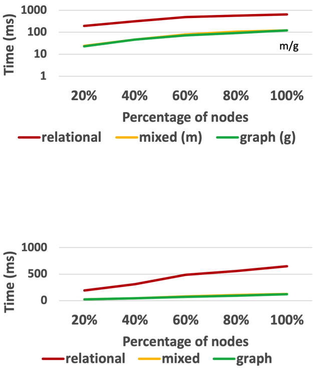


(c) *sf*=1: $S \subset PS \subset L$

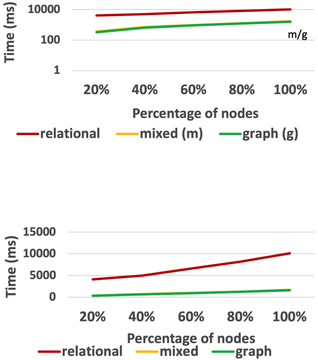
How well does update propagation work under different semantics? - 3/4



(a) $sf{=}0.01$: $C \subset O$

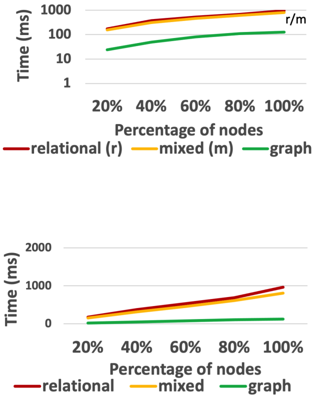


(b) $sf{=}0.1$: $C \subset O$

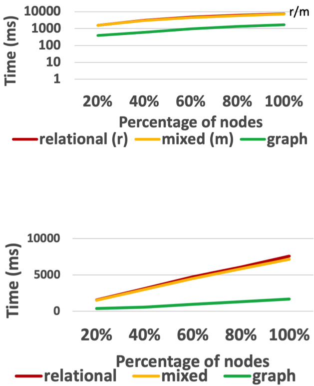


(c) $sf{=}1$: $C \subset O$

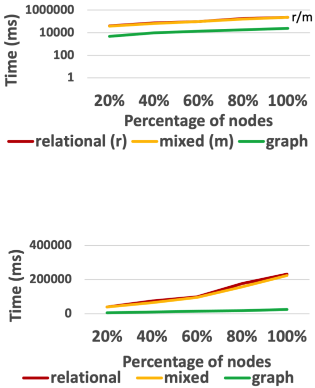
How well does update propagation work under different semantics? - 4/4



(a) *sf*=0.01: $O \subset L$



(b) *sf*=0.1: $O \subset L$



(c) *sf*=1: $O \subset L$

Quantifying the effort of update propagation in TPC-H

chain	semantics	*sf*=0.01		*sf*=0.1		*sf*=1	
		db hits	time (ms)	db hits	time (ms)	db hits	time (ms)
P_c	relational	1,514,378	1,251	16,315,447	12,579	151,030,378	559,684
	mixed	42,002	87	420,002	932	4,200,002	11,689
	graph	6,001	17	60,001	200	600,001	2,022
S_c	relational	29,072,628	18,996	290,157,279	231,131	2,904,674,611	2,216,487
	mixed	40,102	76	401,002	925	4,010,002	10,584
	graph	301	1	3,001	8	30,001	78
C_c	relational	106,502	64	1,065,002	648	10,650,002	10,146
	mixed	4,501	12	45,001	123	450,001	1,707
	graph	4,501	11	45,001	122	450,001	1,582
O_c	relational	315,877	964	3,152,862	7,606	31,522,479	231,556
	mixed	315,877	807	3,152,862	7,175	31,522,479	224,179
	graph	45,001	125	450,001	1,682	4,500,001	23,516

Eliminating property/attribute redundancy is a holy grail in database design

Conclusion

Contributions: "Lean and Mean"

Conceptual modeling

- E/R graphs as semantics of E/R models
- Implementation of E/R modeling within a native technology platform

Contributions: "Lean and Mean"

Conceptual modeling

- E/R graphs as semantics of E/R models
- Implementation of E/R modeling within a native technology platform

Graph modeling

- E/R models as a core fragment of PG-Schema for well-designed (graph) databases
- Property graphs (E/R graph models) as a general data modeling tool

Contributions: "Lean and Mean"

Conceptual modeling

- E/R graphs as semantics of E/R models
- Implementation of E/R modeling within a native technology platform

Graph modeling

- E/R models as a core fragment of PG-Schema for well-designed (graph) databases
- Property graphs (E/R graph models) as a general data modeling tool

Entity and referential integrity management

- E/R keys as efficient fragment of PG-Key for managing integrity of well-designed databases
- Relational, mixed, graph semantics as principled, complementary choices for managing integrity

Contributions: “Lean and Mean”

Conceptual modeling

- E/R graphs as semantics of E/R models
- Implementation of E/R modeling within a native technology platform

Graph modeling

- E/R models as a core fragment of PG-Schema for well-designed (graph) databases
- Property graphs (E/R graph models) as a general data modeling tool

Entity and referential integrity management

- E/R keys as efficient fragment of PG-Key for managing integrity of well-designed databases
- Relational, mixed, graph semantics as principled, complementary choices for managing integrity

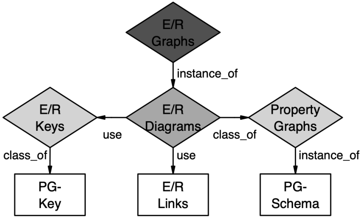
General

- Unifying conceptual, logical and graph data modeling
- Taking integrity management to the next level by eliminating property redundancy

Future Work

Fundamentals

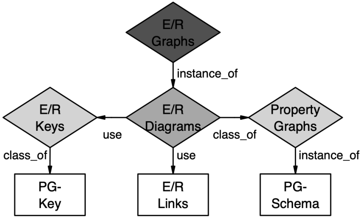
- What are other sweet spots of PG-Schema?
- What are other sweet spots of PG-Key?
- What are classes of well-designed databases?
- How to optimize E/R graph models?
- How to infer models from property graphs?



Future Work

Fundamentals

- What are other sweet spots of PG-Schema?
- What are other sweet spots of PG-Key?
- What are classes of well-designed databases?
- How to optimize E/R graph models?
- How to infer models from property graphs?



Practical

- What does index support for E/R keys and PG-Keys look like?
- What does support for other classes of graph data dependencies look like?

Summary: References

1. Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, Dusan Zivkovic: PG-Schema: Schemas for Property Graphs. Proc. ACM Manag. Data 1(2): 198:1-198:25 (2023)
2. Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Keith W. Hare, Jan Hidders, Victor E. Lee, Bei Li, Leonid Libkin, Wim Martens, Filip Murlak, Josh Perryman, Ognjen Savkovic, Michael Schmidt, Juan F. Sequeda, Slawek Staworko, Dominik Tomaszuk: PG-Keys: Keys for Property Graphs. SIGMOD Conference 2021: 2423-2436
3. Catriel Beeri, Ronald Fagin, David Maier, Mihalis Yannakakis: On the Desirability of Acyclic Database Schemes. J. ACM 30(3): 479-513 (1983)
4. Angela Bonifati, George H. L. Fletcher, Hannes Voigt, Nikolay Yakovets: Querying Graphs. Synthesis Lectures on Data Management, Morgan & Claypool Publishers 2018, ISBN 978-3-031-00736-1
5. Nimo Beeren, George Fletcher: A Formal Design Framework for Practical Property Graph Schema Languages. EDBT 2023: 478-484
6. Peter P. Chen: The Entity-Relationship Model - Toward a Unified View of Data. ACM Trans. Database Syst. 1(1): 9-36 (1976)
7. Mark Levene, Millist W. Vincent: Justification for Inclusion Dependency Normal Form. IEEE Trans. Knowl. Data Eng. 12(2): 281-291 (2000)
8. Heikki Mannila, Kari-Jouko Räihä: Design of Relational Databases. Addison-Wesley 1992, ISBN 0-201-56523-4
9. Bernhard Thalheim: Entity-relationship modeling - foundations of database technology. Springer 2000, ISBN 978-3-540-65470-4, pp. I-XII, 1-627