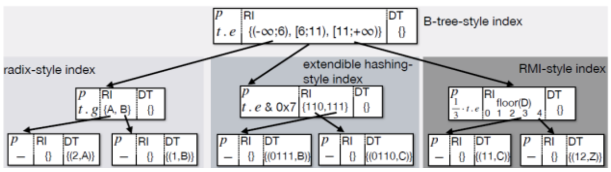
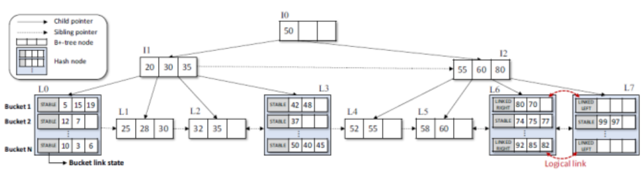
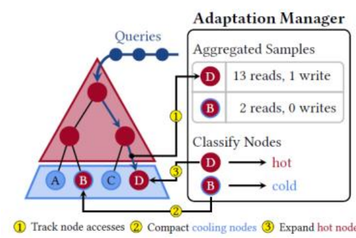
Addendum on Optimizations: Hybrid Indexes



[1]



[2]

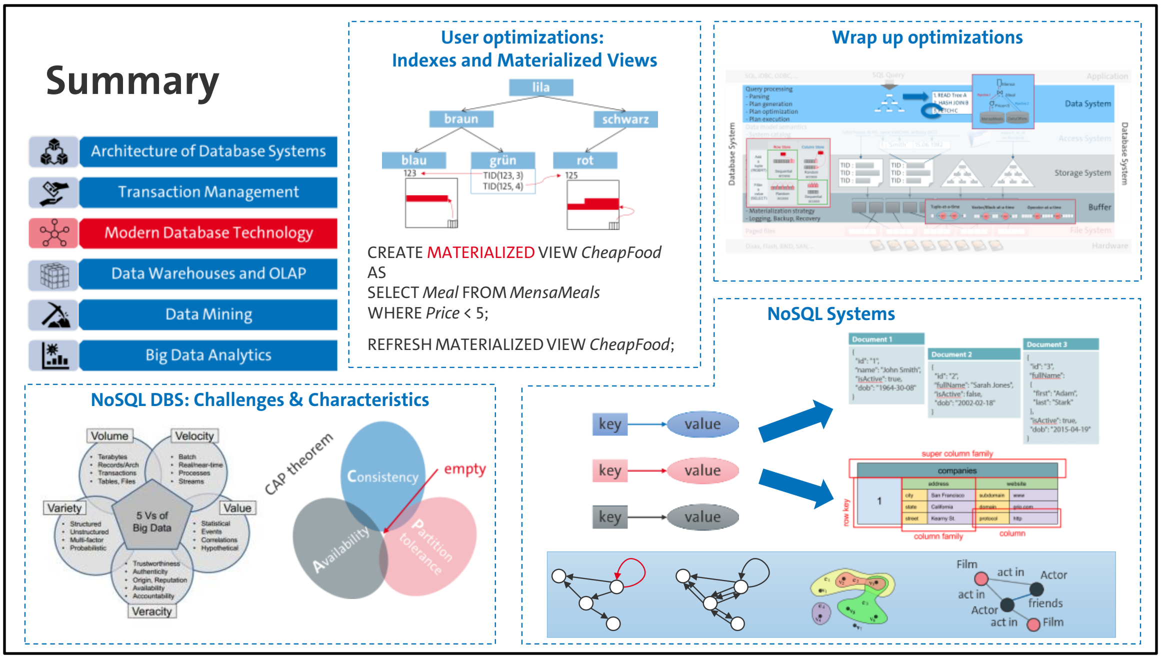


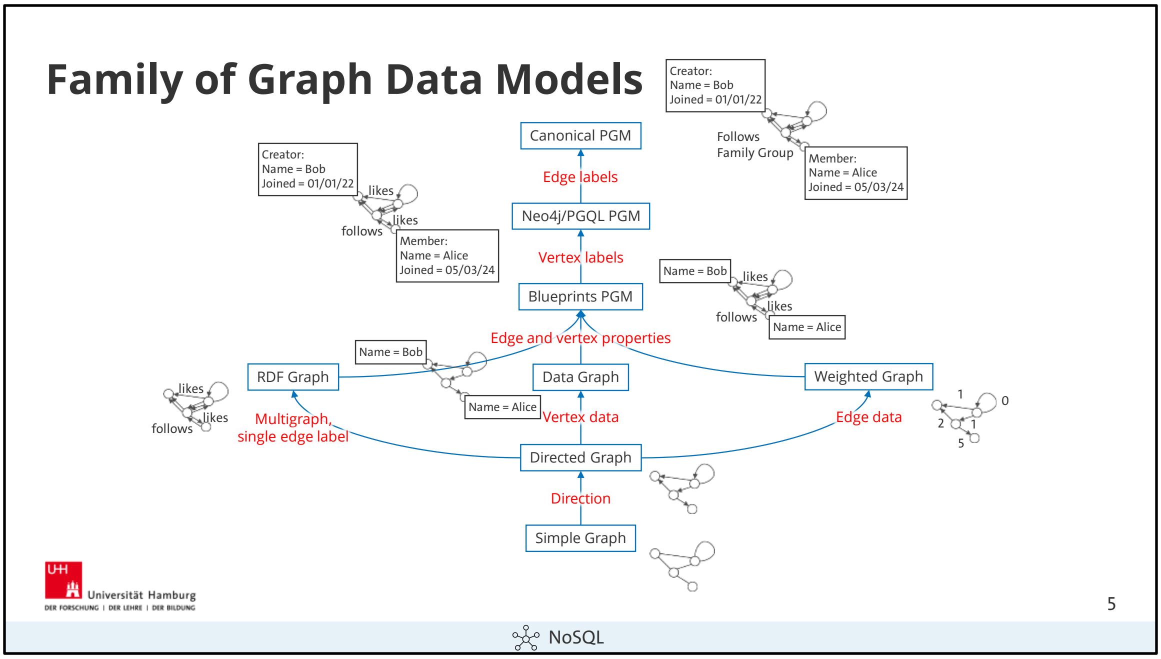
[3]

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Optimizations in RDBMS

[1] Dittrich, Jens, Joris Nix, and Christian Schön. "The next 50 years in database indexing or: the case for automatically generated index structures." *Proceedings of the VLDB Endowment* 15.3 (2021): 527-540.
[2] Cha, Hokeun, et al. "Blink-hash: An adaptive hybrid index for in-memory time-series databases." *Proceedings of the VLDB Endowment* 16.6 (2023): 1235-1248.
[3] Anneser, Christoph, et al. "Adaptive hybrid indexes." *Proceedings of the 2022 International Conference on Management of Data*. 2022.


Summary
Architecture of Database Systems
Transaction Management
Modern Database Technology
Data Warehouses and OLAP
Data Mining
Big Data Analytics
User optimizations:
Indexes and Materialized Views
lila
braun
schwarz
blau
grün
rot
123
TID(123, 3)
TID(125, 4)
125
CREATE MATERIALIZED VIEW CheapFood
AS
SELECT Meal FROM MensaMeals
WHERE Price < 5;
REFRESH MATERIALIZED VIEW CheapFood;
Wrap up optimizations
Data System
Storage System
Buffer
Database System
NoSQL Systems
Document 1
Document 2
Document 3
key
value
super column family
companies
address
website
row key
column family
column
NoSQL DBS: Challenges & Characteristics
Volume
Velocity
Variety
Value
Veracity
5 Vs of Big Data
CAP theorem
Consistency
Availability
Partition tolerance
empty
Film
act in
Actor
friends



PGM - Property Graph Model
RDF - Resource Description Framework
→ Stored in triple format (node, edge label, node)
→ Each property must be modeled as a triple Label: Descriptive Type information
PGQL – Property Graph Query Language (tries to look like SQL)
Property/Data - Actual data
Label – Descriptive (type) information
→ Labels are sometimes called “type”, especially in the context of edges

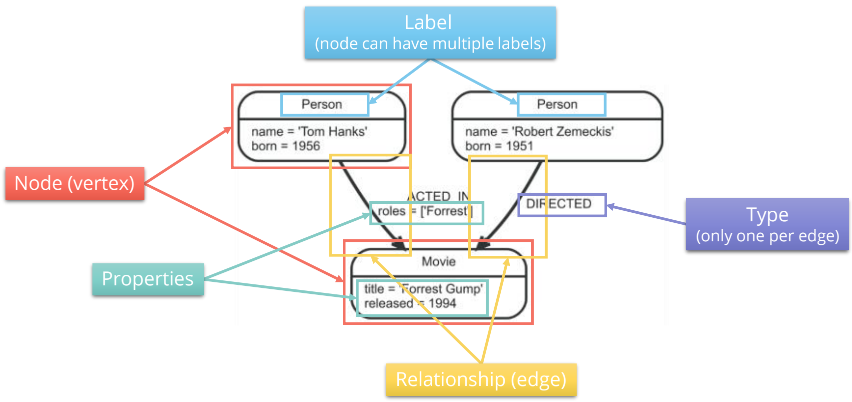
Properties/Data:

- Assuming a domain of data / values $\mathcal{D}$ and a domain of property keys / attributes $\mathcal{K}$
- A graph is a structure (V, E, ν)
- $\nu: V \rightarrow \mathcal{D}$ (vertex data)
  $\nu: E \rightarrow \mathcal{D}$ (edge data)
  $\nu: V \cup E \rightarrow \mathcal{D}$ (vertex and edge data)
  $\nu: V \times \mathcal{K} \rightarrow \mathcal{D}$ (vertex properties)
  $\nu: E \times \mathcal{K} \rightarrow \mathcal{D}$ (edge properties)

$\nu: (V \cup E) \times \mathcal{K} \to \mathcal{D}$ (vertex and edge properties)
is a partial function assigning data / values to each
vertex / edge / vertex and edge

Neo4j PGM

[http://neo4j.com/docs/developer-manual/current/introduction/#graphdb-concepts]

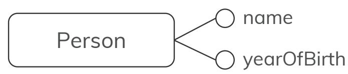


Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Property Graph Modelling

Entity with Properties

- Entity relationship model

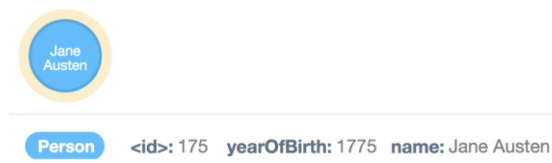


- Schema typically implicit, i.e. given with instances

```
(ja:Person { name: 'Jane Austen', yearOfBirth: 1775 })
```

- Instance

Person
name="Jane Austen"
yearOfBirth=1775



Vertex id is system generated in Neo4j

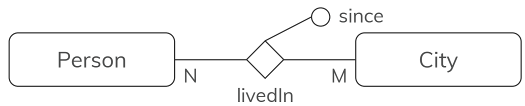
Cypher (Neo4j) uses the keyword CREATE to create a new vertex, e.g.

CREATE (ja:Person { name: 'Jane Austen', yearOfBirth: 1775 })

Property Graph Modelling

Relationships (N:M)

- Entity relationship model

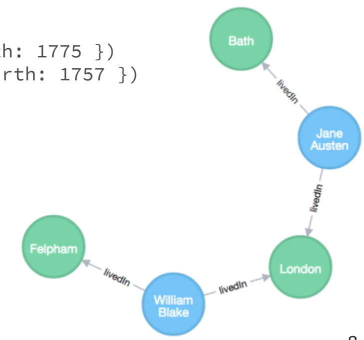


- Vertices

```
(ja:Person { name: 'Jane Austen', yearOfBirth: 1775 })
(wb:Person { name: 'William Blake', yearOfBirth: 1757 })
(lo:City {name: 'London'})
(ba:City {name: 'Bath'})
(fe:City {name: 'Felpham'})
```

- Edges

```
(ja)-[:livedIn {since: 1775}]->(lo)
(ja)-[:livedIn {since: 1800}]->(ba)
(wb)-[:livedIn {since: 1757}]->(lo)
(wb)-[:livedIn {since: 1800}]->(fe)
```

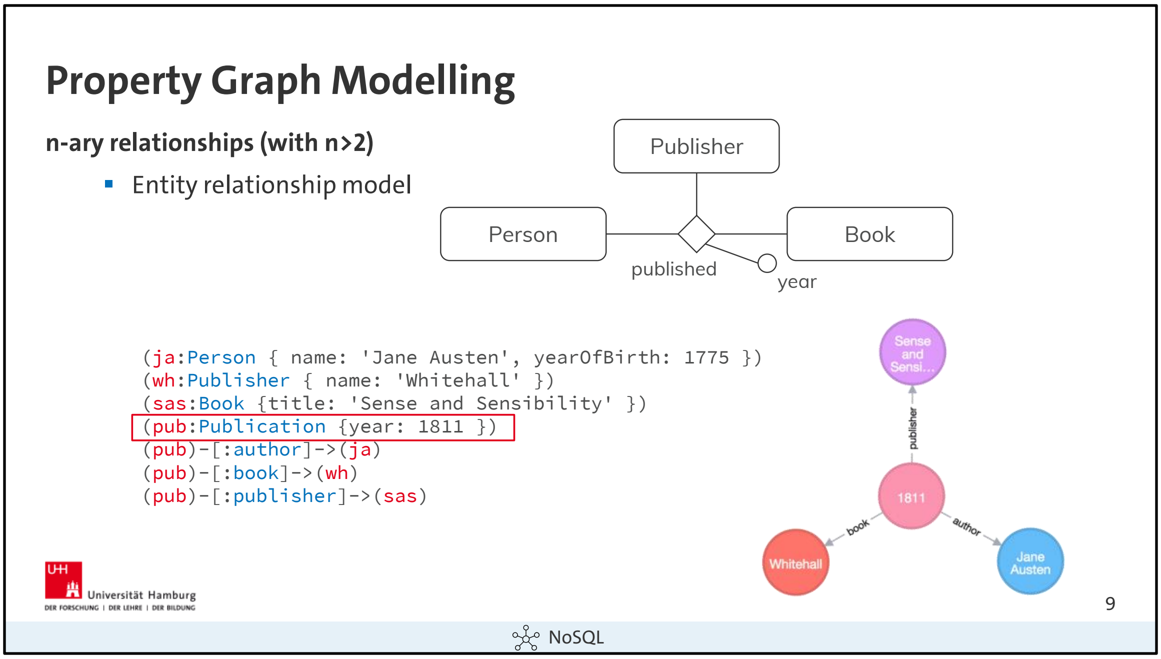


Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

8

NoSQL

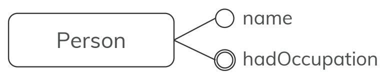
An edge can be created with the CREATE keyword in Cypher, just like vertices, e.g.
CREATE (ja)-[:livedIn {since: 1775}]->(lo)



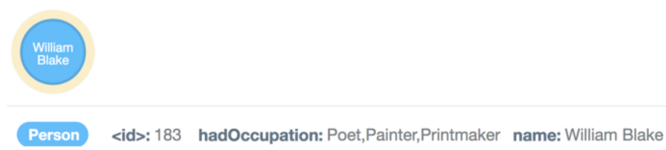
Not directly expressible since edge connect exactly two vertices, not more, not less
→ Solution: Reification of relationship

Property Graph Modelling

Multivalued properties




```
(wb:Person {name: 'William Blake', hadOccupation: ['Poet','Painter','Printmaker']})
```



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

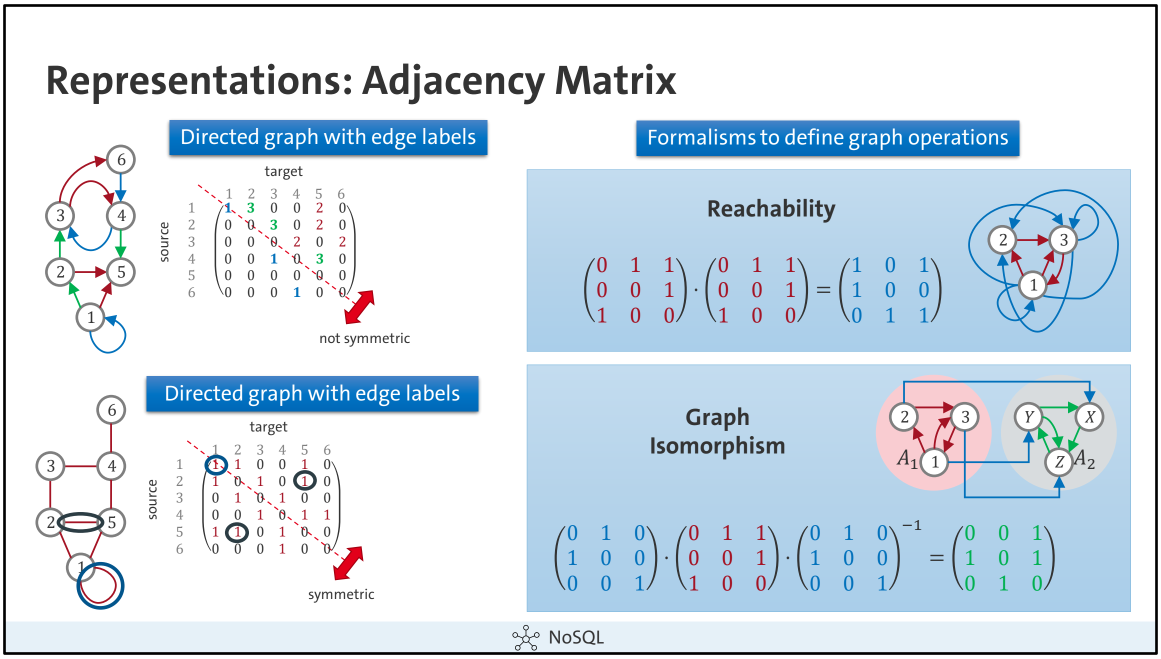
Expressibility depends on datatype system of specific system in use
→ Possible in Neo4j (see slide)
→ If not expressible: Reification of values of multivalued property

Multiple labels can be connected with a : or a & in Neo4j, e.g. (wb:Person&Artist {name: 'William Blake', hadOccupation: ['Poet','Painter','Printmaker']})

Exercise Neo4j

- Your fridge is stocked with cheese, butter, milk, and pizza
- There are also 4 tomatoes and two bars of chocolate in the kitchen
- Each item has an expiration date and a quantity: cheese (15 May, 10), butter (01 June, 1), milk (11 May, 2), Pizza (25 Aug, 5)
- You have a recipe for luxury frozen pizza that is called “PiDeLuxe”. It requires a slice of cheese and a tomato per frozen pizza. You want to prepare one Pizza “PiDeLuxe” today.

How can this be modeled using the ER model?
How could it be expressed with Neo4j?

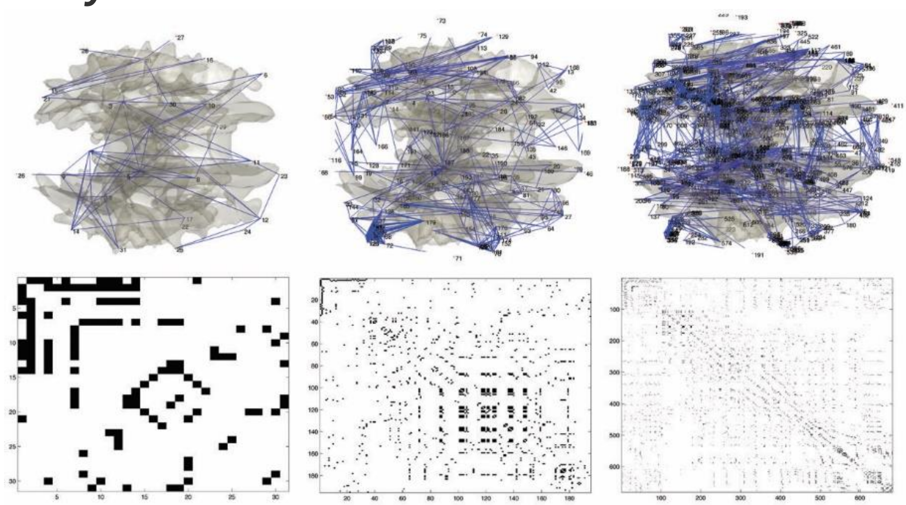


Reachability: $A^2 = A \cdot A$ shows all paths of length 2 between nodes of graph represented by A

Graph isomorphism: Graphs represented by A_1 and A_2 are isomorphic if there exists a permutation matrix P such that $PA_1\, P^{-1}$=A_2

Note: Most definitions work only for unlabeled graphs!

Adjacency Matrix



[http://brainimaging.waisman.wisc.edu/~chung/graph/admatrix.jpg]

14

NoSQL

Adjacency matrices are often sparse
→ Compression

Compress Sparse Row (CSR)

$$
\begin{array}{c c}
 & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} a & & c & \\ & b & & e \\ d & i & f & \\ & h & g & \end{pmatrix}
\end{array}
$$

Position for row # in other two arrays:

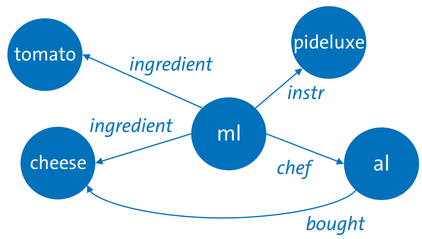
	#	0	1	2	3
Row position array:		0	2	4	7

	#	0	1	2	3	4	5	6	7	8
Column index array:		0	2	1	3	0	1	2	1	2
Cell value array:		a	c	b	e	d	i	f	h	g

15

Also works column-wise → Compress Sparse Column (CSC)

Exercise CSR



What does the adjacency matrix for this graph look like?
Apply Compress Sparse Row.

Representations: Adjacency List

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & a & & c & \\ 1 & & b & & e \\ 2 & d & i & f & \\ 3 & & h & g & \end{array}$$

Source vertex with outgoing edges...

...without edge labels	...with edge labels	...with edge properties
0 -> (0,2)	0 -> ([0,a],[2,c])	0 -> ([0,a,(weight=4)],[2,c,(weight=3)])
1 -> (1,3)	1 -> ([1,b],[3,e])	1 -> ([1,b,(weight=3)],[3,e,(weight=2)])
2 -> (0,1,2)	2 -> ([0,d],[1,i],[2,f])	2 -> ([0,d,(weight=5)],[1,i,(weight=2)],...)
3 -> (1,2)	3 -> ([1,h],[2,g])	3 -> ([1,h,(weight=9)],[2,g,(weight=7)])

The same!

Almost the same!

Compressed Sparse Row

0	2	4	7

0	2	1	3	0	1	2	1	2
a	c	b	e	d	i	f	h	g

source-oriented

Compressed Sparse Column

0	2	5	8

0	2	1	2	3	0	2	3	1
a	d	b	i	h	c	f	g	e

target-oriented

Representations: Relational Representations I

Triple Table: A table with three columns: subject, predicate, object

Subject	Predicate	Object
<http://www.w3.org/People/EM/contact#me>	<http://www.w3.org/2000/10/swap/pim/contact#fullName>	"Eric Miller"
<http://www.w3.org/People/EM/contact#me>	<http://www.w3.org/2000/10/swap/pim/contact#mailbox>	<mailto:e.miller123(at)example>
<http://www.w3.org/People/EM/contact#me>	<http://www.w3.org/2000/10/swap/pim/contact#personalTitle>	"Dr."
<http://www.w3.org/People/EM/contact#me>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.w3.org/2000/10/swap/pim/contact#Person>

Vertex and Edge Table: Two universal tables, one for vertices, one for edges

ID	Type	Color	Name	RAM	Nationality
1	Product	black	"Apple iPad MC707LL/A"	64 GB	
…	…	…	…	…	…
4	Category		"Cell Phones & Accessories"		
5	Category		"Phones"		
…	…	…	…	…	…

Source	Target	Type	Rating
1	7	in	
…	…	…	…
5	4	part of	
7	6	part of	
…	…	…	…

19

NoSQL

Triple Table

- Generic, simple, straightforward approach
- Can be built on relational storage engines
- Add indexing for faster lookups
- Queries need to operate on all triples (no data portioning based on schema information)
- High storage redundancy

Vertex and Edge Table

- Requires efficient handling of NULL values
    - Given for instance in column stores
- Queries need to operate on all vertices/edges (no data portioning based on schema information)

Further reading

Daniel J. Abadi et al. Scalable Semantic Web Data Management Using Vertical Partitioning. VLDB 2007

Representations: Relational Representations II

Clustered Property Table: Groups properties that tend to be defined together in a table

Property Table

Subj.	Type	Title	copyright
ID1	BookType	"XYZ"	"2001"
ID2	CDType	"ABC"	"1985"
ID3	BookType	"MNP"	NULL
ID4	DVDType	"DEF"	NULL
ID5	CDType	"GHI"	"1995"
ID6	BookType	NULL	"2004"

Left-Over Triples

Subj.	Prop.	Obj.
ID1	author	"Fox, Joe"
ID2	artist	"Orr, Tim"
ID2	language	"French"
ID3	language	"English"

Property-Class Table: Cluster similar sets of subjects together in the same table

Class: BookType

Subj.	Title	Author	copyright
ID1	"XYZ"	"Fox, Joe"	"2001"
ID3	"MNP"	NULL	NULL
ID6	NULL	NULL	"2004"

Class: CDType

Subj.	Title	Artist	copyright
ID2	"ABC"	"Orr, Tim"	"1985"
ID5	"GHI"	NULL	"1995"

Left-Over Triples

Subj.	Prop.	Obj.
ID2	language	"French"
ID3	language	"English"
ID4	type	DVDType
ID4	title	"DEF"

Clustered Property Table

- Example:
    - Properties: type, title, and copyright date
    - Table stores all triples with one of these properties
- Multiple property tables with different clusters of properties may be created
- One particular property may only appear in at most one property table
- Triples that not fit any property table are stored in a left-over triple table
- Multivalued attributes are problematic
- Queries that have unspecified property are problematic
- Introduce NULL values

Property-Class Table

- Exploits the type property of subjects
- A property may exist in multiple property-class tables
- A subject may also exist in multiple tables
- Multivalued attributes are problematic
- Queries that do not select on class type are problematic
- Introduce NULL values

Property Table Approaches

Triple Table

Subj.	Prop.	Obj.
ID1	type	BookType
ID1	title	"XYZ"
ID1	author	"Fox, Joe"
ID1	copyright	"2001"
ID2	type	CDType
ID2	title	"ABC"
ID2	artist	"Orr, Tim"
ID2	copyright	"1985"
ID2	language	"French"
ID3	type	BookType
ID3	title	"MNO"
ID3	language	"English"
ID4	type	DVDType
ID4	title	"DEF"
ID5	type	CDType
ID5	title	"GHI"
ID5	copyright	"1995"
ID6	type	BookType
ID6	copyright	"2004"

(Clustered) property table

Property Table

Subj.	Type	Title	copyright
ID1	BookType	"XYZ"	"2001"
ID2	CDType	"ABC"	"1985"
ID3	BookType	"MNP"	NULL
ID4	DVDType	"DEF"	NULL
ID5	CDType	"GHI"	"1995"
ID6	BookType	NULL	"2004"

Left-Over Triples

Subj.	Prop.	Obj.
ID1	author	"Fox, Joe"
ID2	artist	"Orr, Tim"
ID2	language	"French"
ID3	language	"English"

Property-Class Table

Class: BookType

Subj.	Title	Author	copyright
ID1	"XYZ"	"Fox, Joe"	"2001"
ID3	"MNP"	NULL	NULL
ID6	NULL	NULL	"2004"

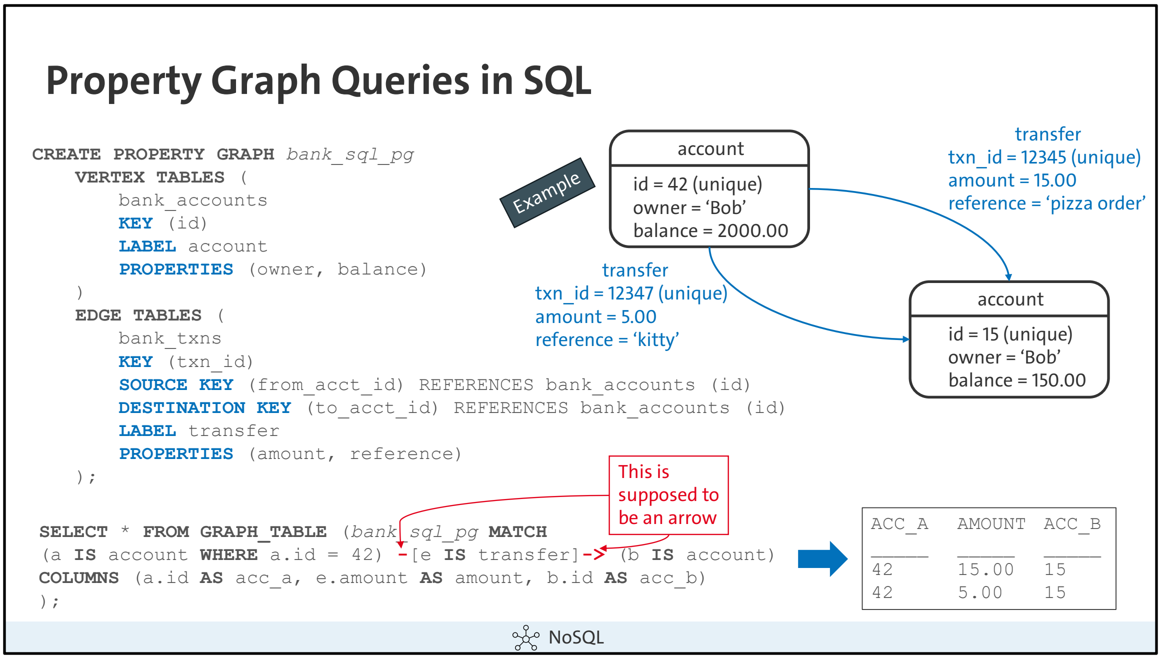
Class: CDType

Subj.	Title	Artist	copyright
ID2	"ABC"	"Orr, Tim"	"1985"
ID5	"GHI"	NULL	"1995"

Left-Over Triples

Subj.	Prop.	Obj.
ID2	language	"French"
ID3	language	"English"
ID4	type	DVDType
ID4	title	"DEF"

Reduce numbers of subject-subject self joins necessary to reconstruct entities



- PGQs were recently adopted in the SQL standard (2023): ISO/IEC DIS 9075-16
- First implementation in Oracle version 23.2: https://docs.oracle.com/en/database/oracle/property-graph/23.2/spgdg/sql-property-graphs.html
- **Only because PGQs are supported by SQL, there is no guarantee that the DBS implementing it is optimized for graph processing!**