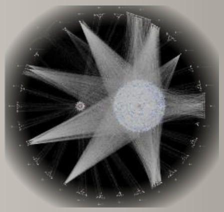
Construction and Training of Multi-Associative Graph Networks

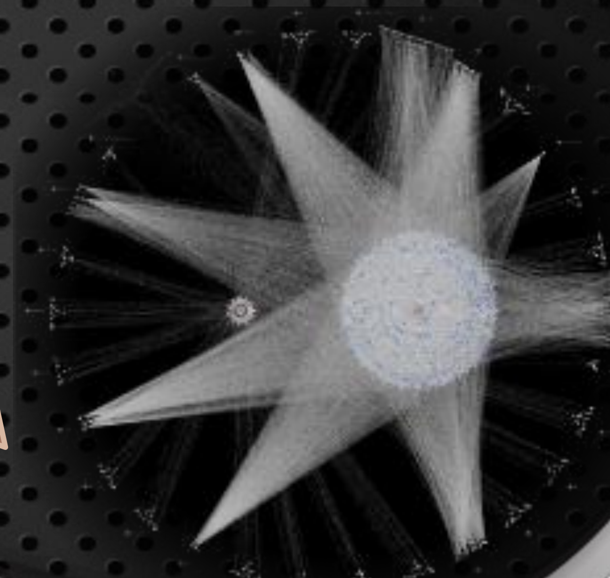


Adrian Horzyk
Daniel Bulanda
Janusz A. Starzyk

AGH University of Krakow - University of Information Technology and Management in Rzeszow - Ohio University in Athens

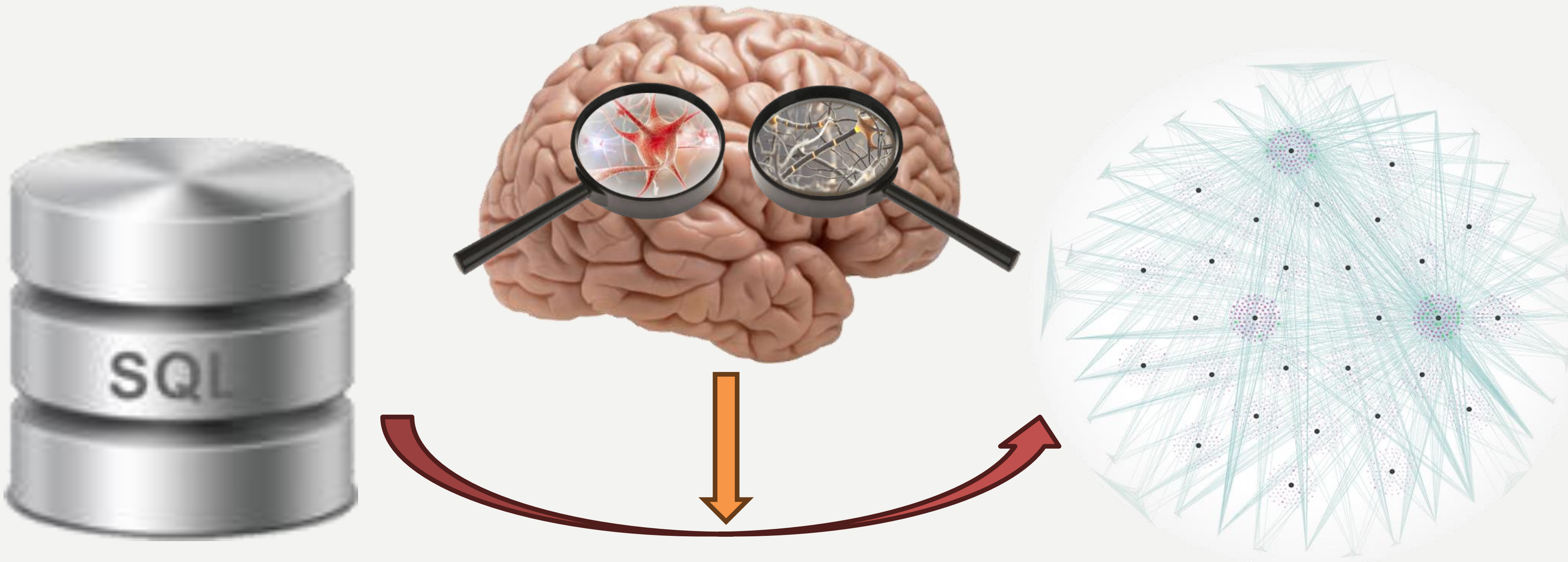
ECML PKDD 2023

Aim and Objectives



Construction and Training of Multi-Associative Graph Networks

Aim and Objectives



The goal of this research is to

understand how data and relationships are represented in the brains

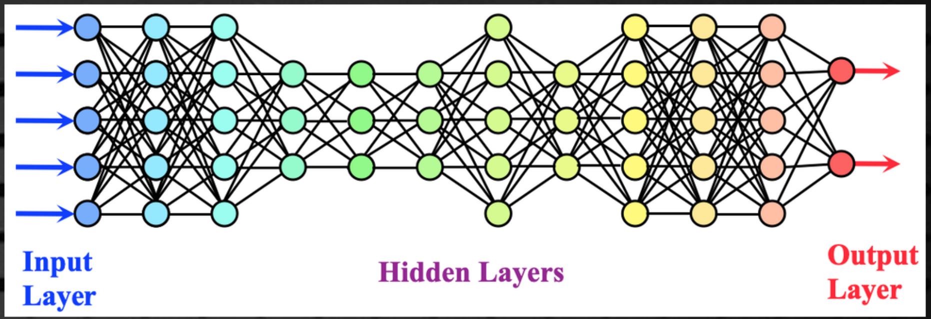
and use this information to represent multiple data of any kind

in graph-based structures that behave like neural networks

to represent knowledge about the data, objects, and their relations

to use it for solving different computational intelligence tasks

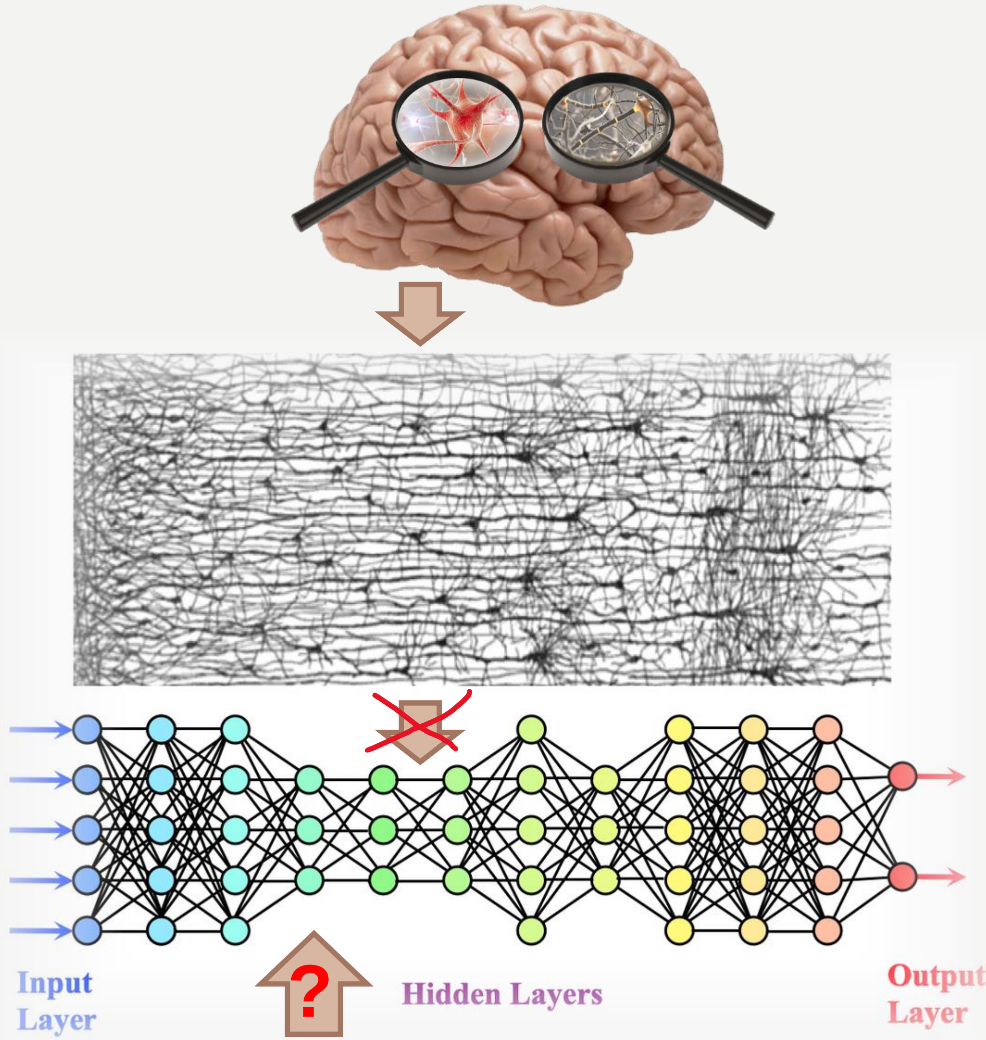
efficiently without predefined targets before „training”.



How do brains differ from contemporary neural networks?

Construction and Training of Multi-Associative Graph Networks

Essential Differences



Brains contain sparse, irregular, lateral, recurrent, adaptable, and dynamically developed connections during the life-long training process, reproducing diverse relationships between represented objects by neurons,

while

contemporary NN structures are usually rigid, regular, layered, and feed-forward with predefined inputs and outputs containing regular all-to-all or many-to-many area-limited connections that can adapt only weights.

How to use data stored in various databases efficiently in brain-like manner?

Insufficient DB representation of data relationships

RDB Relationship Representation

Data stored in RDBs are usually imperfectly normalized and aggregated to avoid too many join operations that impact search efficiency.

table: **doctors**

id_doc	first_name	last_name	experience	specialization
D1	John	Smith	25	Surgery
D2	Tom	Allen	10	Radiology
D3	David	Jolie	45	Surgery
D4	John	Smith	8	Oncology
D5	Raul	Willis	2	Patology
D6	Lucy	Hanks	25	Radiology
D7	Nina	Ford	10	Surgery
D8	Tom	Cage	2	Surgery

link table: **doctors-patients**

id_doc	id_pat
D2	P4
D7	P4
D2	P6
D7	P6
D2	P7
D4	P7
D6	P7
D7	P7
D5	P7
D5	P1
D5	P2
D1	P3
D3	P3
D8	P5
D7	P5
D1	P8
D8	P8
D3	P9
D7	P9
D2	P2
D2	P5
D6	P8
D6	P9

table: **patients**

id_pat	first_name	last_name	age	disease	room_id
P1	John	Cage	35	Lupus	R4
P2	John	Smith	58	Lupus	R1
P3	Emma	Hanks	42	Colitis	R3
P4	Nina	Ford	70	Pneumonia	R4
P5	Lucy	Allen	70	Colitis	R3
P6	Sara	Ford	25	Pneumonia	R4
P7	Cate	Bosch	62	Glioma	R3
P8	David	Smith	42	Infarct	R1
P9	Raul	Willis	25	Infarct	R2

table: **nurses**

id_nur	first_name	last_name	experience
N1	Cate	Bosch	25
N2	Emma	Allen	8
N3	Nina	Smith	10
N4	Sara	Ford	2

link table: **nurses-rooms**

id_nur	id_rom
N1	R3
N1	R4
N2	R3
N2	R4
N3	R1
N3	R2
N4	R1
N4	R2

table: **rooms**

id_rom	type	beds
R1	PostOp	3
R2	Emergency	2
R3	Ward	4
R4	Ward	5

Many relationships require the use of queries to find them.

RDBs represent only a small part of useful relationships, while similarity, order, and the same values of attributes of objects stored in the same or different tables are unrepresented.

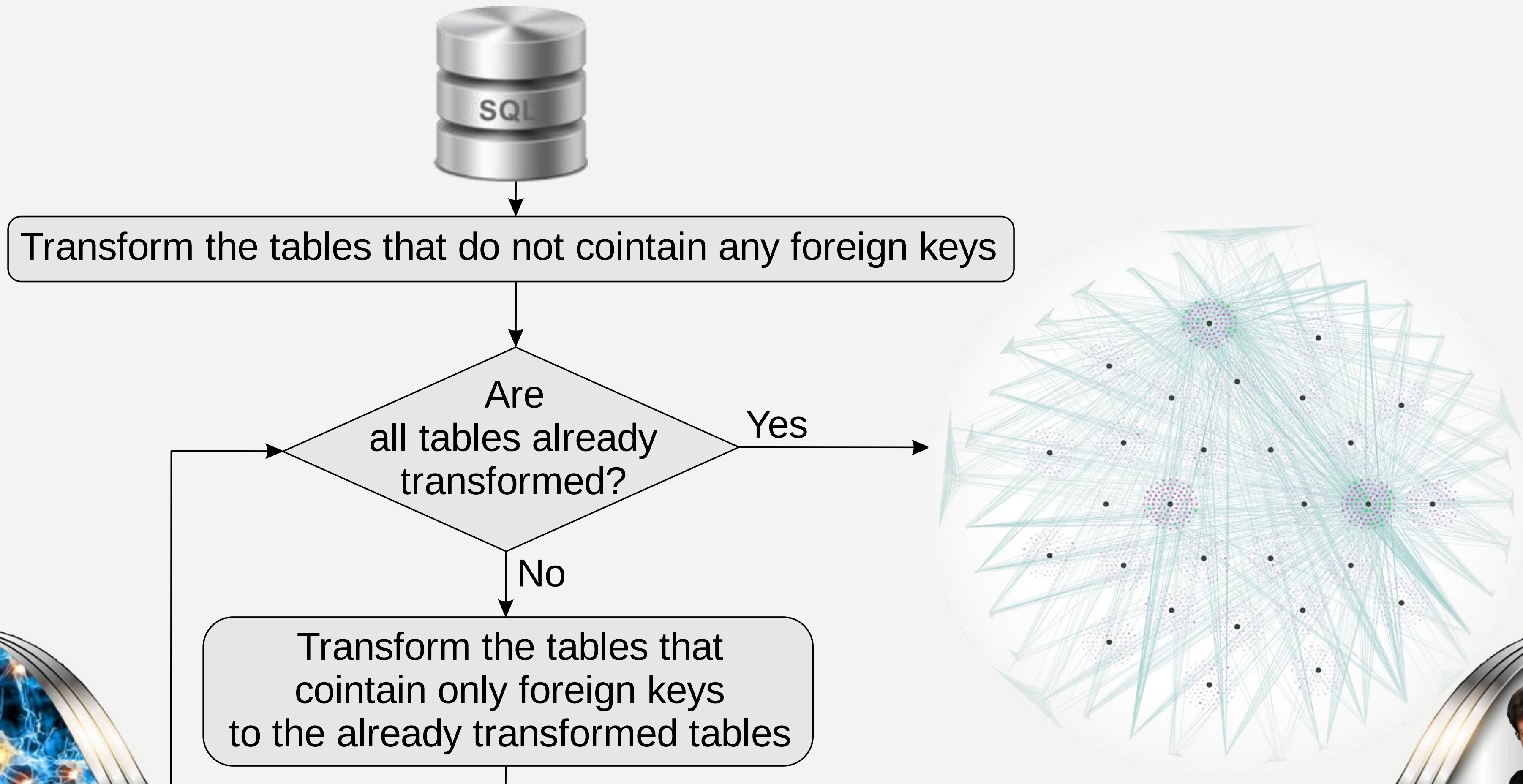
ECML PKDD 2023

Associative DB Transformation to a Multi-Associative Graph Network

Construction and Training of Multi-Associative Graph Networks

Associative Transformation Algorithm

Let's transform all Relational Database (RDB) tables to the graph structure aggregating all the same values in the same nodes, connecting all neighbor values in order, joining all tables' columns of all RDBs representing the same categories, following the presented algorithm:



Example of Associative Transformation

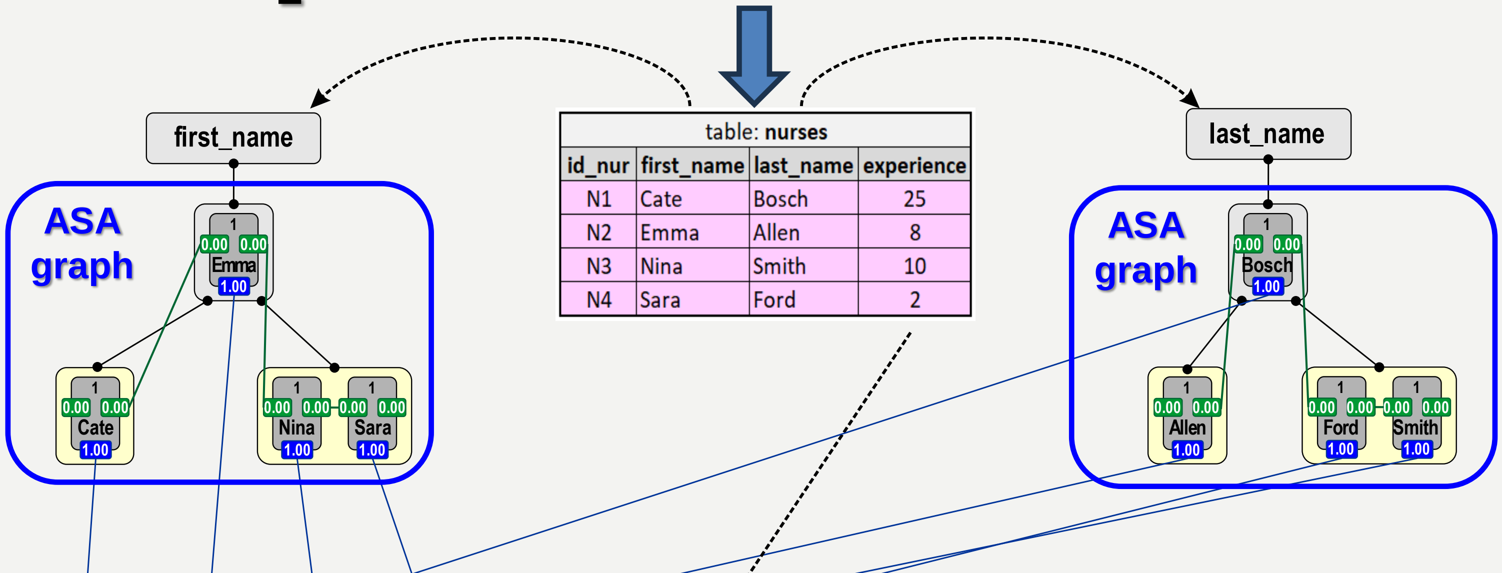
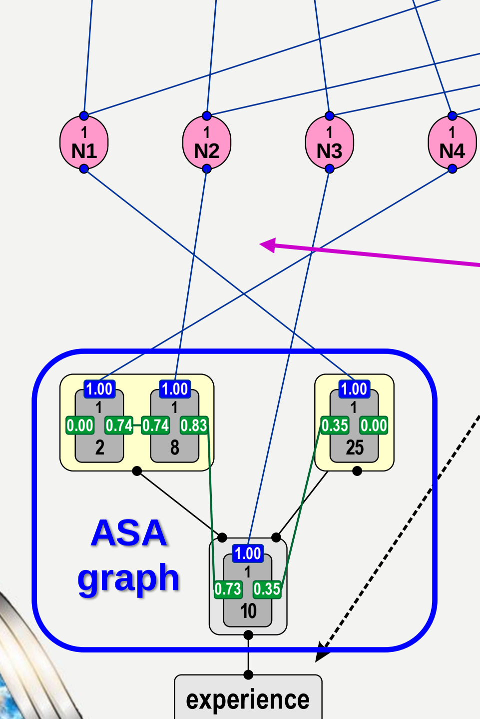


table: nurses			
id_nur	first_name	last_name	experience
N1	Cate	Bosch	25
N2	Emma	Allen	8
N3	Nina	Smith	10
N4	Sara	Ford	2



We start the **database associative transformation process** from table „nurses” because it contains **no foreign keys** to any other tables.

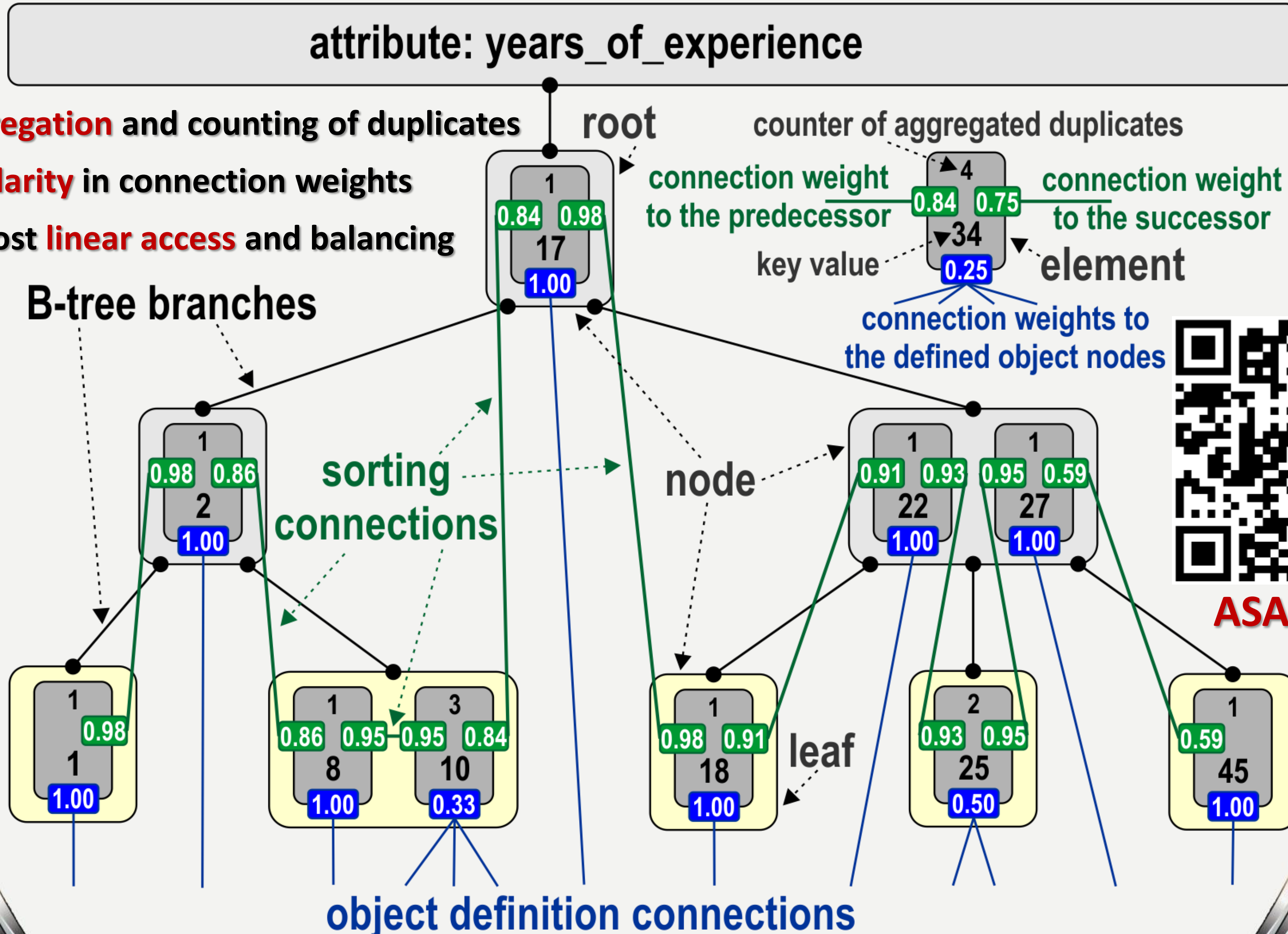
Each table row (entity) is represented by **a node** that is connected to the nodes representing its defining values (and other entities).

Simple attribute values are stored in the **ASA-graphs** for better performance.

ASA-graph

ASA-graph is an associative self-sorting and self-balancing B-tree-based graph structure for representing attribute values in the sorted order:

- **Aggregation and counting of duplicates**
- **Similarity in connection weights**
- **Almost linear access and balancing**



ASA-graph

Example of Associative Transformation

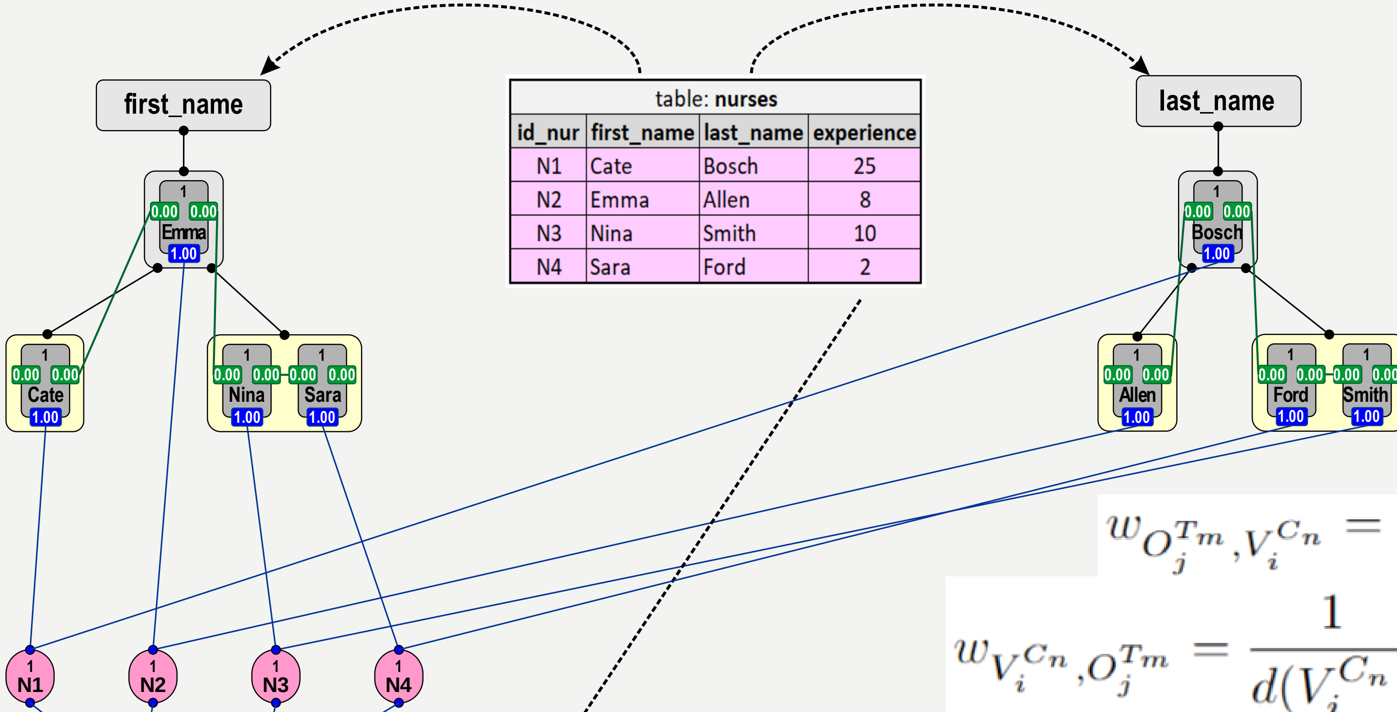
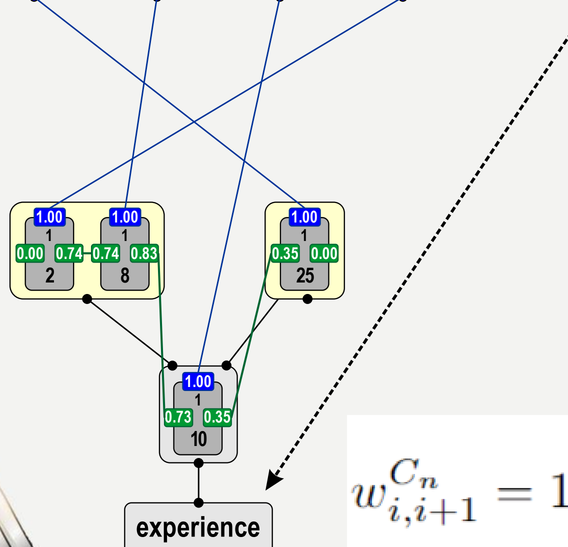


table: nurses			
id_nur	first_name	last_name	experience
N1	Cate	Bosch	25
N2	Emma	Allen	8
N3	Nina	Smith	10
N4	Sara	Ford	2



$$w_{O_j^{T_m}, V_i^{C_n}} = 1$$

$$w_{V_i^{C_n}, O_j^{T_m}} = \frac{1}{d(V_i^{C_n})}$$

All connections are weighted.

The weights are calculated according to the similarity of values represented by the nodes or the number of represented values or objects:

$$w_{O_k^{T_n}, O_j^{T_m}} = \frac{1}{d(O_k^{T_n})}$$

$$w_{i,i+1}^{C_n} = 1 - \frac{(v_{i+1}^{C_n} - v_i^{C_n})}{R^{C_n}}$$

$$R^{C_n} = v_{max}^{C_n} - v_{min}^{C_n}$$

Example of Associative Transformation

Sparse graph connections work like hard attention.

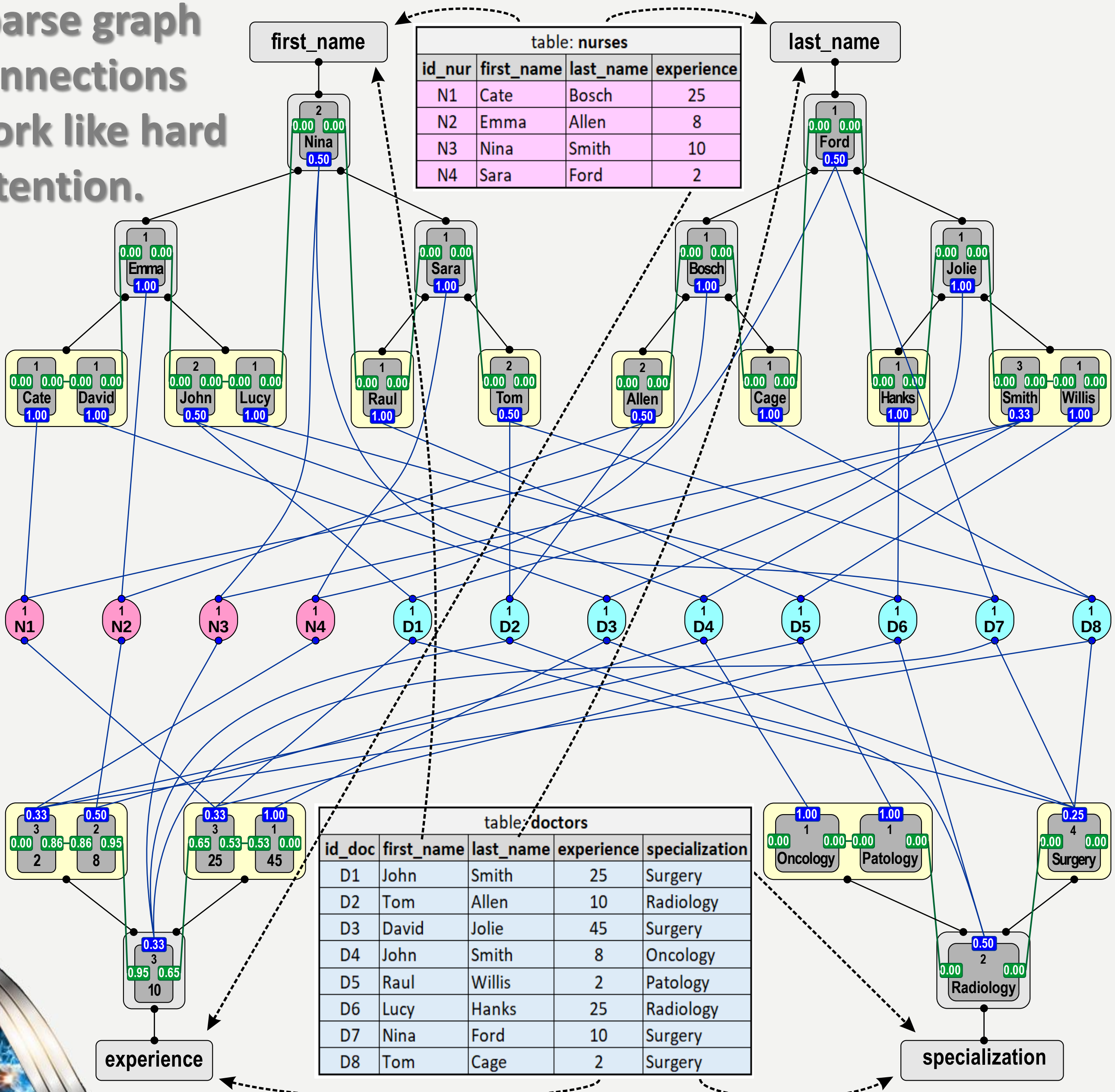


table: nurses

id_nur	first_name	last_name	experience
N1	Cate	Bosch	25
N2	Emma	Allen	8
N3	Nina	Smith	10
N4	Sara	Ford	2

table: doctors

id_doc	first_name	last_name	experience	specialization
D1	John	Smith	25	Surgery
D2	Tom	Allen	10	Radiology
D3	David	Jolie	45	Surgery
D4	John	Smith	8	Oncology
D5	Raul	Willis	2	Patology
D6	Lucy	Hanks	25	Radiology
D7	Nina	Ford	10	Surgery
D8	Tom	Cage	2	Surgery

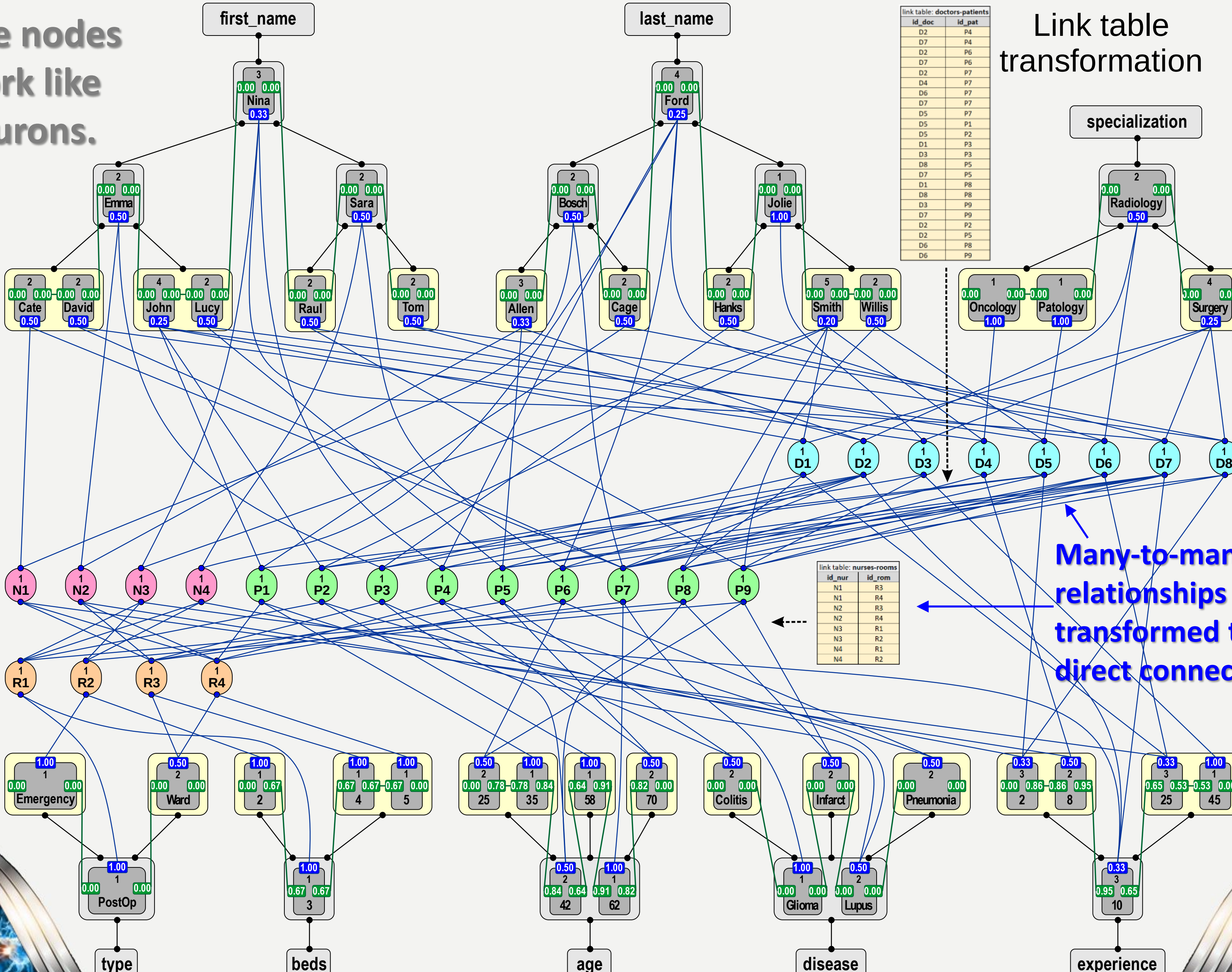
In the next steps, the other tables are transformed and their values are aggregated in the same ASA-graphs if belonging to the same data categories, e.g.: „first_name”, „last_name”, or „experience”, and counters and weights are updated.

Example of Associative Transformation

The nodes work like neurons.

Link table transformation

Many-to-many relationships are transformed to direct connections.



link table: doctors-patients	
id_doc	id_pat
D2	P4
D7	P4
D2	P6
D7	P6
D2	P7
D4	P7
D6	P7
D7	P7
D5	P7
D5	P1
D5	P2
D1	P3
D3	P3
D8	P5
D7	P5
D1	P8
D8	P8
D3	P9
D7	P9
D2	P2
D2	P5
D6	P8
D6	P9

link table: nurses-rooms	
id_nur	id_rom
N1	R3
N1	R4
N2	R3
N2	R4
N3	R1
N3	R2
N4	R1
N4	R2

MAGN Constructed for RDB

table: patients

id_pat	first_name	last_name	age	disease	room_id
P1	John	Cage	35	Lupus	4
P2	John	Smith	58	Lupus	1
P3	Emma	Hanks	42	Colitis	3
P4	Nina	Ford	70	Pneumonia	4
P5	Lucy	Allen	70	Colitis	3
P6	Sara	Ford	25	Pneumonia	4
P7	Cate	Bosch	62	Glioma	3
P8	David	Smith	42	Infarct	1
P9	Raul	Willis	25	Infarct	2

table: doctors

id_doc	first_name	last_name	experience	specialization
D1	John	Smith	25	Surgery
D2	Tom	Allen	10	Radiology
D3	David	Jolie	45	Surgery
D4	John	Smith	8	Oncology
D5	Raul	Willis	2	Patology
D6	Lucy	Hanks	25	Radiology
D7	Nina	Ford	10	Surgery
D8	Tom	Cage	2	Surgery

link table: doctors-patients

id_doc	id_pat
D2	P4
D7	P4
D2	P6
D7	P6
D2	P7
D4	P7
D6	P7
D7	P7
D5	P7
D5	P1
D5	P2
D1	P3
D3	P3
D8	P5
D7	P5
D1	P8
D8	P8
D3	P9
D7	P9
D2	P2
D2	P5
D6	P8
D6	P9

link table: nurses-rooms

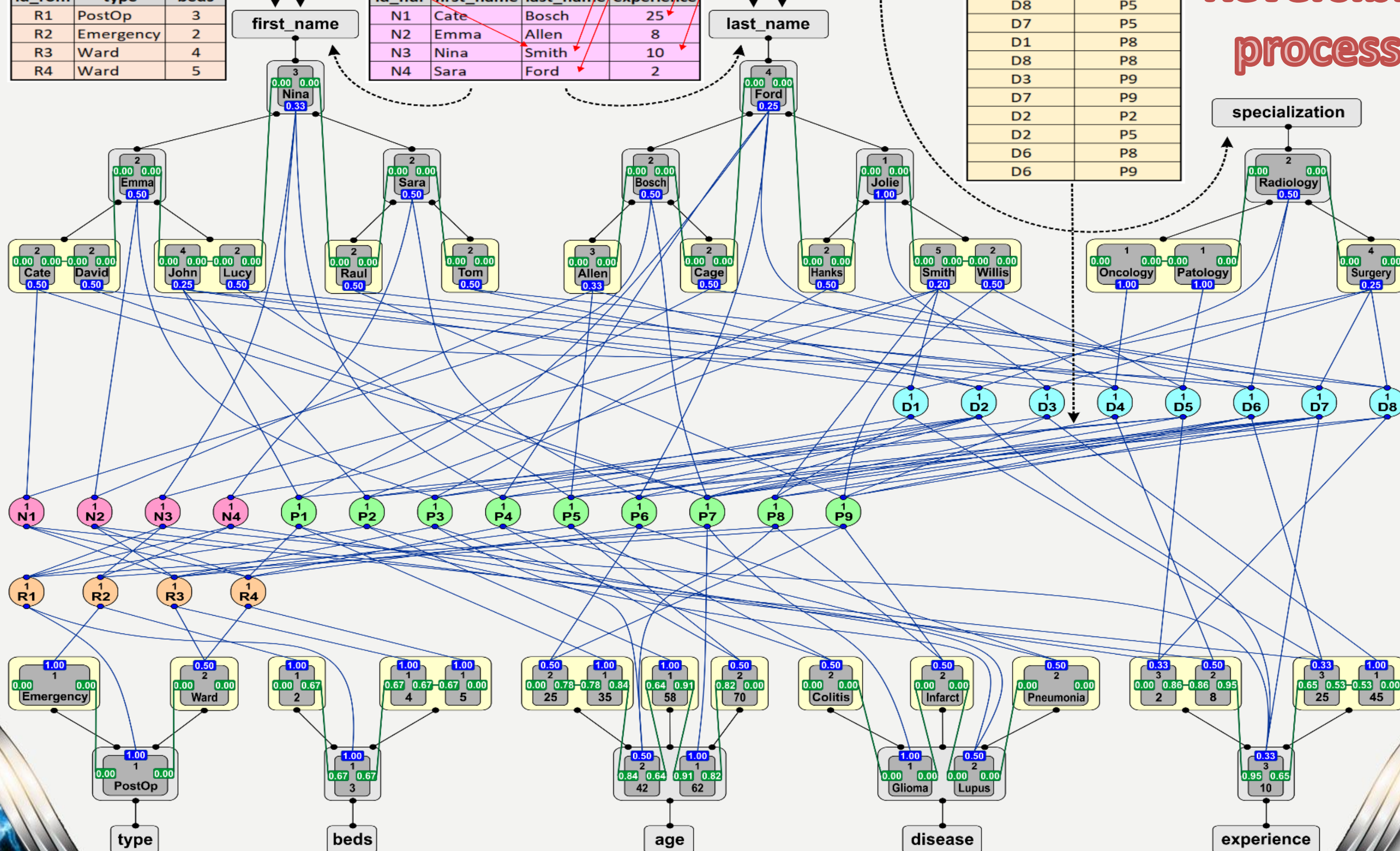
id_nur	id_rom
N1	R3
N1	R4
N2	R3
N2	R4
N3	R1
N3	R2
N4	R1
N4	R2

table: rooms

id_rom	type	beds
R1	PostOp	3
R2	Emergency	2
R3	Ward	4
R4	Ward	5

table: nurses

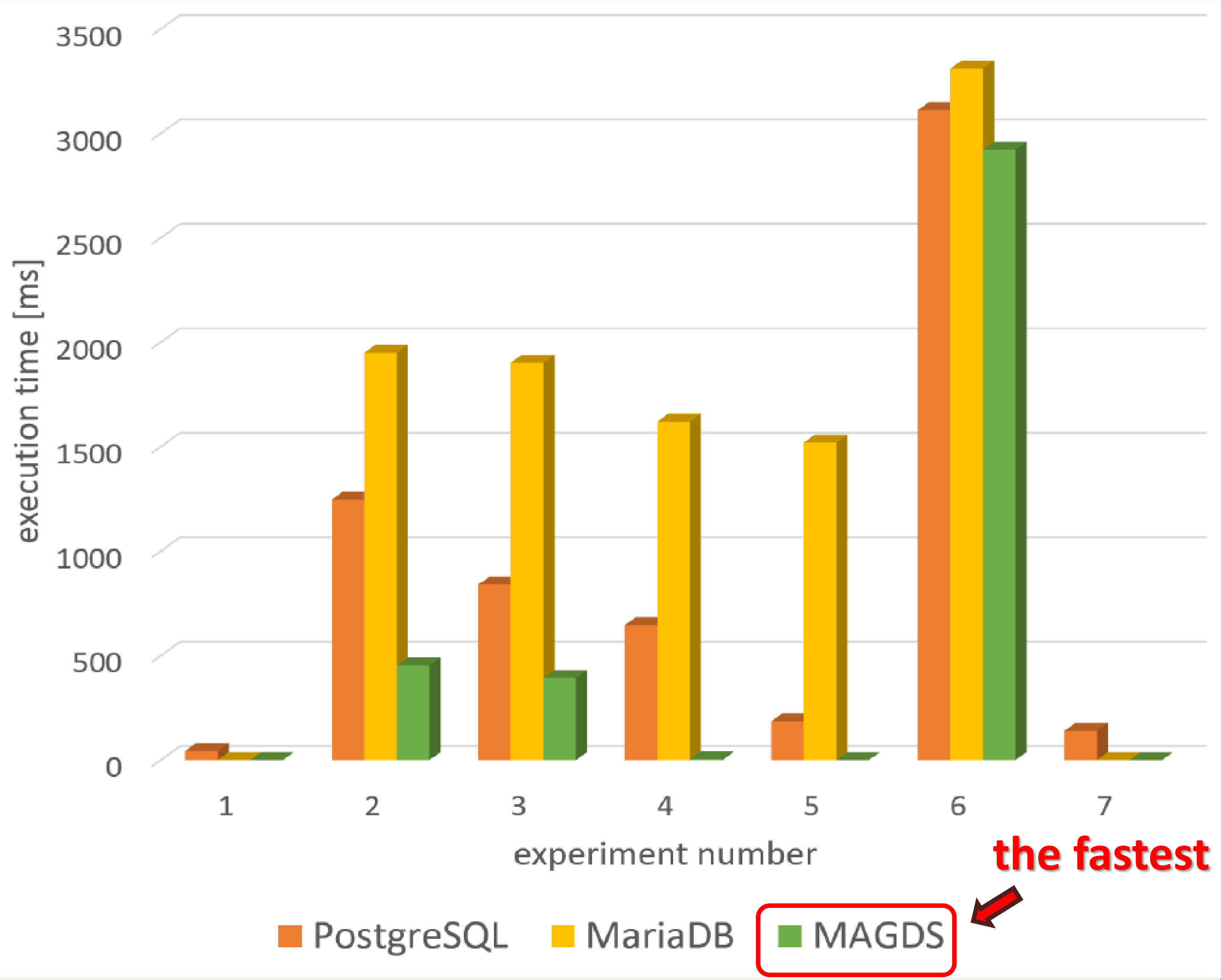
id_nur	first_name	last_name	experience
N1	Cate	Bosch	25
N2	Emma	Allen	8
N3	Nina	Smith	10
N4	Sara	Ford	2



Reversible process

Efficiency of MAGNs

Comparisons of SQL query times of MAGNs and RDBMs in milliseconds:



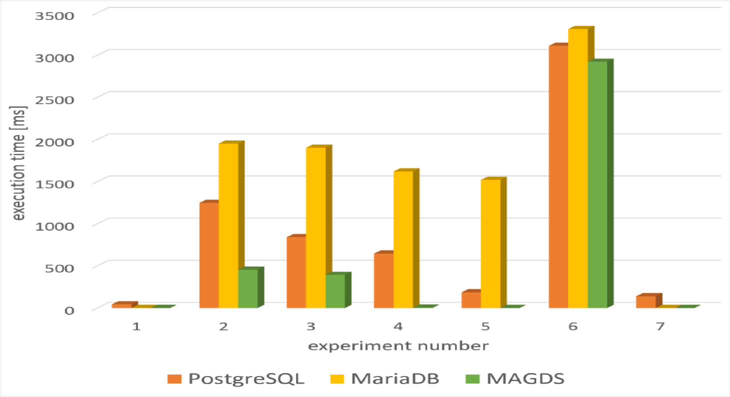
Queries Used in the Comparisons

Experiment 1

select age from gxd_specimen limit 100;

Experiment 2

select distinct sp.specimenlabel
from gxd_specimen sp
inner join gxd_genotype ge
on sp._genotype_key = ge._genotype_key
inner join prb_strain st
on ge._strain_key = st._strain_key
inner join mgi_user us
on st._createdby_key = us._user_key
where agemin in (select min(agemin) from gxd_specimen)
or agemin in (select max(agemin) from gxd_specimen)
and agemax in (select min(agemax) from gxd_specimen)
or agemax in (select max(agemax) from gxd_specimen)
and ge.isconditional = 0
and st.standard = 1
and st.private = 0
order by specimenlabel;



Experiment 3

select sum(sp.sequencenum) / count(sp.sequencenum)
from gxd_specimen sp
inner join gxd_genotype ge
on sp._genotype_key = ge._genotype_key
inner join prb_strain st
on ge._strain_key = st._strain_key
inner join mgi_user us
on st._createdby_key = us._user_key
where agemin in (select min(agemin) from gxd_specimen)
or agemin in (select max(agemin) from gxd_specimen)
and agemax in (select min(agemax) from gxd_specimen)
or agemax in (select max(agemax) from gxd_specimen)
and ge.isconditional = 0
and st.standard = 1
and st.private = 0;

Experiment 4

select count(distinct insertsize) from prb_probe;

Experiment 5

select max(insertsize) from prb_probe;

Experiment 6

select
sum(distinct startcoordinate) / count(distinct startcoordinate)
from map_coord_feature;

Experiment 7

select count(*) from map_coord_feature;

Example of another RDB

This DB contains **multiple and missing values** (DB is denormalized).

Database

table: **patient**

id	person_id	room_id	ICD-11
1	9	1	['CA01', 'BA00']
2	10	1	['BD11', '2F7A']
3	11	3	['EA80']
4	13	2	['EA80', 'BA00']
5	12	4	['BD11', 'EA80']
6	15	4	['2F7A', 'BA00', 'SA10']
7	16	2	['SA10']
8	14	4	['BA00', 'EA80', 'SA11']

table: **doctor**

id	person_id	patient_id	specialization	years_of_experience
1	2	1	Surgery	25
2	1	2	Radiology	10
3	4	4	Surgery	45
4	3	5	Oncology	17
5	18	3	Patology	2
6	20		Surgery	22
7	17	7	Radiology	25
8	19	8	Surgery	10
9	21	6	Surgery	1

table: **room**

id	type	no_beds
1	PostOp	3
2	Emergency	3
3	Ward	4
4	Ward	5

table: **nurse**

id	person_id	room_id	years_of_experience
1	7	4	10
2	5	3	1
3	8	1	25
4	6	2	2

table: **person**

id	first_name	last_name	age
1	Nina	Musk	35
2	John	Smith	58
3	Evelyn	Musk	42
4	John	Bush	70
5	Evelyn	Ford	70
6	Evelyn	Woods	25
7	Sara	Bosch	62
8	Cate	Smith	35
9	Lucy	Allen	70
10	Cate	Bosch	62
11	Sara	Ford	25
12	David	Smith	42
13	John	Bosch	35
14	David	Musk	58
15	Sara	Hanks	42
16	Nina	Ford	70
17	Nina	Smith	42
18	Lucy	Bush	25
19	David	Allen	35
20	Raul	Hanks	42
21	Raul	Woods	24

MAGNs can manage all these cases without any problems.

Example of another RDB

This DB contains multiple and missing values (DB is denormalized).

table: **patient**

id	person_id	room_id	ICD-11
1	9	1	['CA01', 'BA00']
2	10	1	['BD11', '2F7A']
3	11	3	['EA80']
4	13	2	['EA80', 'BA00']
5	12	4	['BD11', 'EA80']
6	15	4	['2F7A', 'BA00', 'SA10']
7	16	2	['SA10']
8	14	4	['BA00', 'EA80', 'SA11']

Database

table: **person**

id	first_name	last_name	age
1	Nina	Musk	35
2	John	Smith	58
3	Evelyn	Musk	42
4	John	Bush	70
5	Evelyn	Ford	70
6	Evelyn	Woods	25
7	Sara	Bosch	62
8	Cate	Smith	35
9	Lucy	Allen	70
10	Cate	Bosch	62
11	Sara	Ford	25
12	David	Smith	42
13	John	Bosch	35
14	David	Musk	58
15	Sara	Hanks	42
16	Nina	Ford	70
17	Nina	Smith	42
18	Lucy	Bush	25
19	David	Allen	35
20	Raul	Hanks	42
21	Raul	Woods	24

table: **room**

id	type	no_beds
1	PostOp	3
2	Emergency	3
3	Ward	4
4	Ward	5

table: **doctor**

id	person_id	patient_id	specialization	years_of_experience
1	2	1	Surgery	25
2	1	2	Radiology	10
3	4	4	Surgery	45
4	3	5	Oncology	17
5	18	3	Patology	2
6	20		Surgery	22
7	17	7	Radiology	25
8	19	8	Surgery	10
9	21	6	Surgery	1

table: **nurse**

id	person_id	room_id	years_of_experience
1	7	4	10
2	5	3	1
3	8	1	25
4	6	2	2

MAGNs can manage all these cases without any problems.

Construction Step 1

Transformed tables: **Rooms**

Associative transformation of table „rooms”

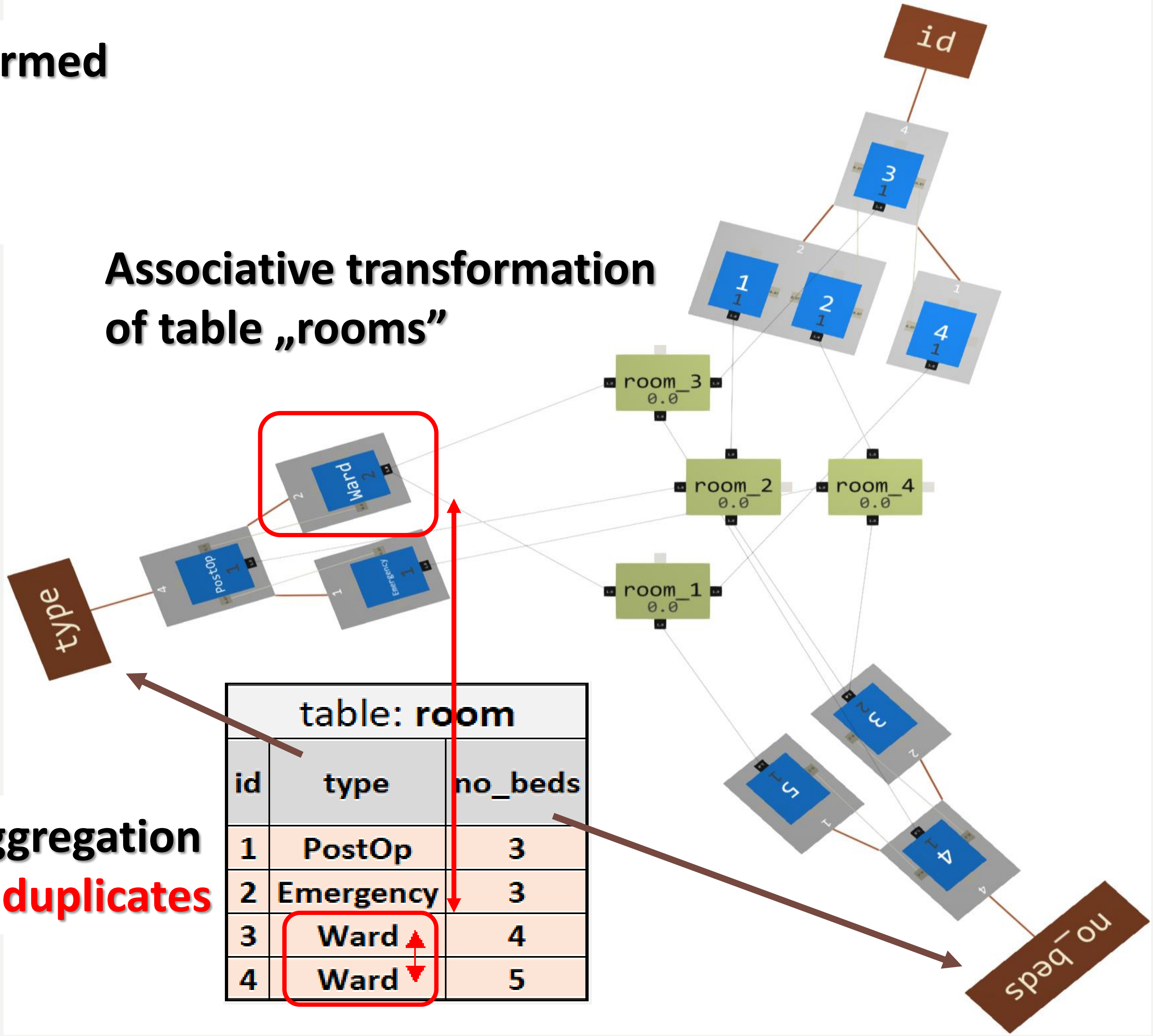


table: room		
id	type	no_beds
1	PostOp	3
2	Emergency	3
3	Ward	4
4	Ward	5

aggregation of duplicates

Construction Step 2

Transformed tables:

Rooms

↓

Persons

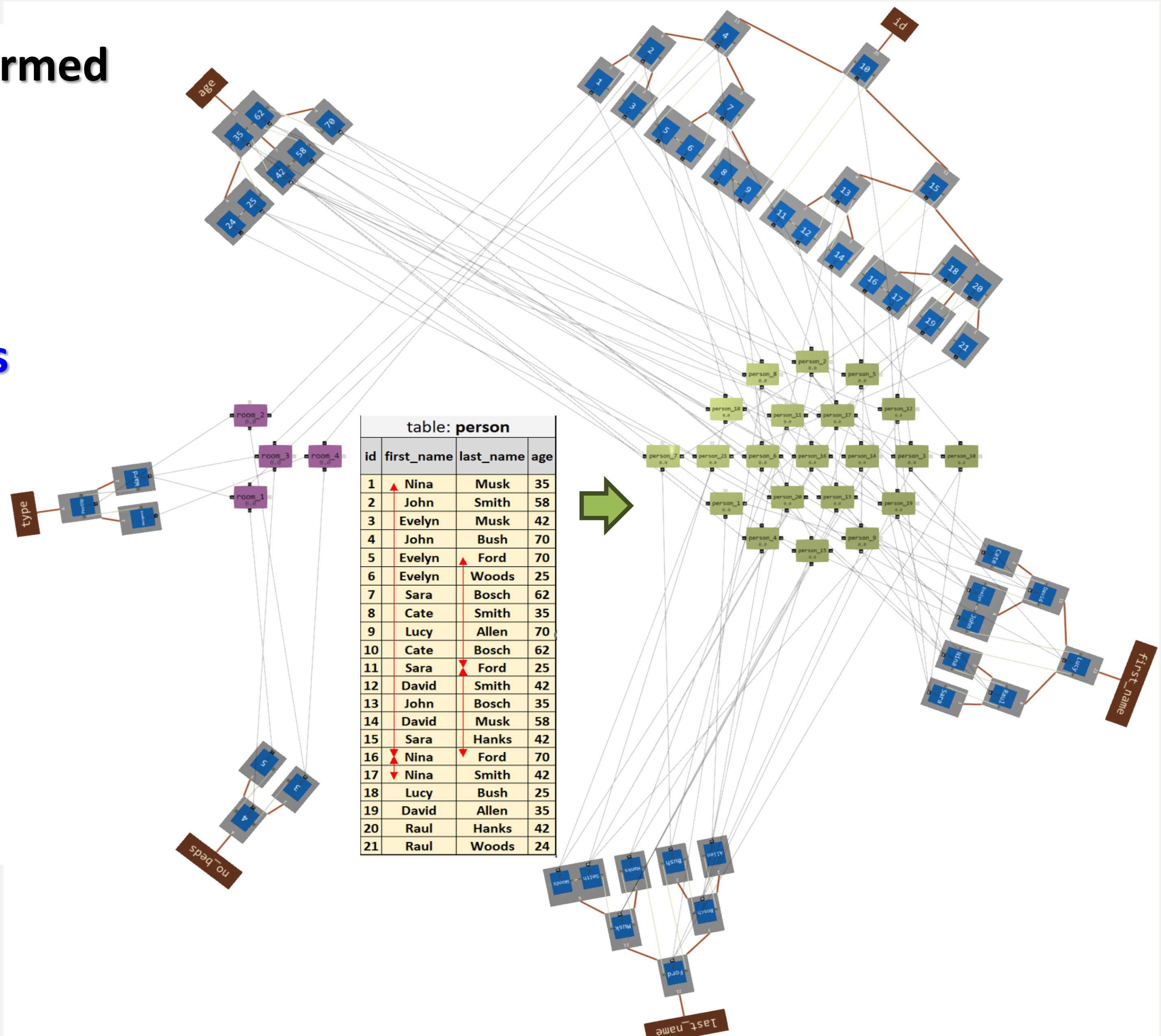


table: **person**

id	first_name	last_name	age
1	Nina	Musk	35
2	John	Smith	58
3	Evelyn	Musk	42
4	John	Bush	70
5	Evelyn	Ford	70
6	Evelyn	Woods	25
7	Sara	Bosch	62
8	Cate	Smith	35
9	Lucy	Allen	70
10	Cate	Bosch	62
11	Sara	Ford	25
12	David	Smith	42
13	John	Bosch	35
14	David	Musk	58
15	Sara	Hanks	42
16	Nina	Ford	70
17	Nina	Smith	42
18	Lucy	Bush	25
19	David	Allen	35
20	Raul	Hanks	42
21	Raul	Woods	24

Construction Step 3

Transformed tables:

Rooms

↓

Persons

↓

Patients

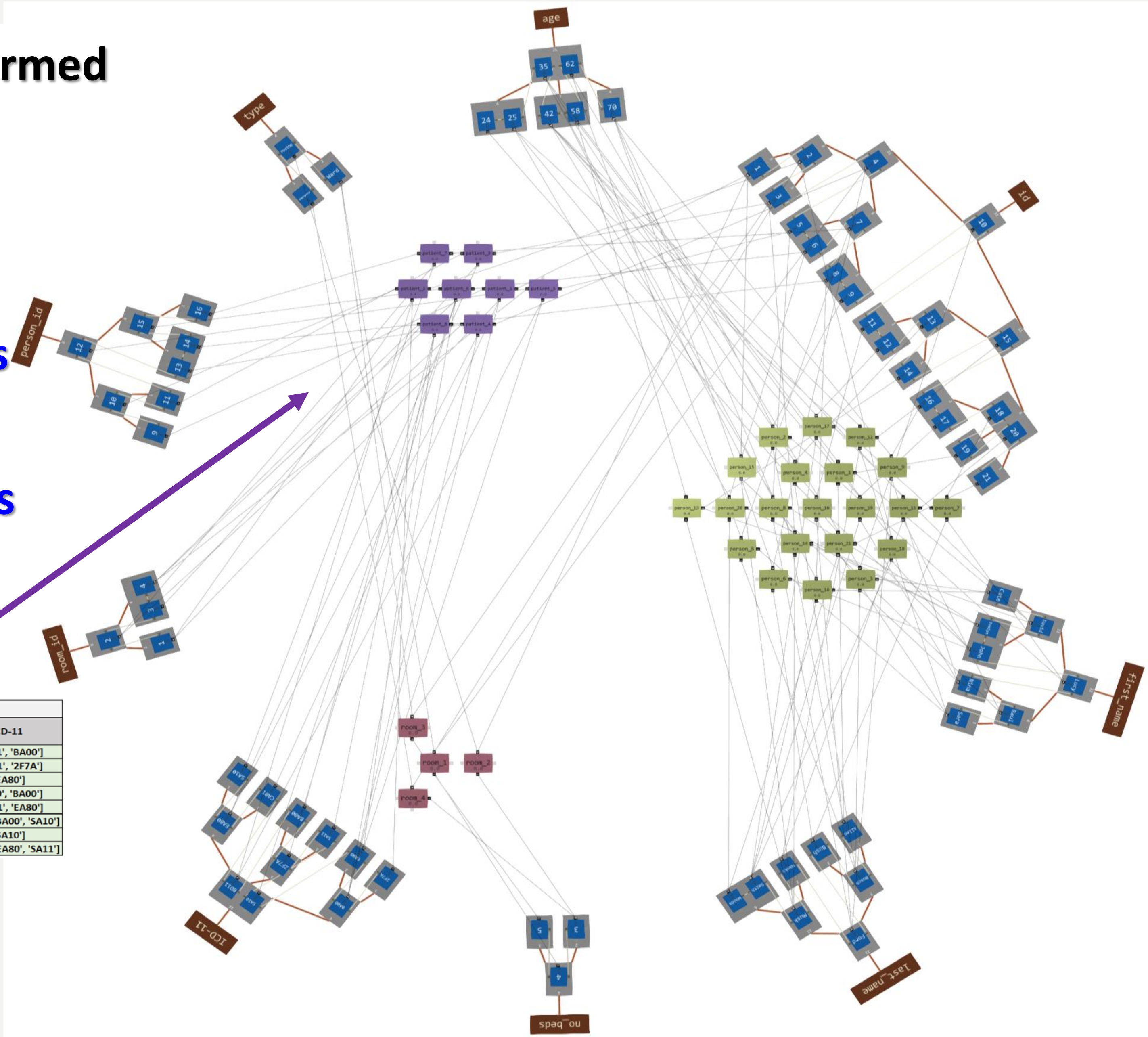


table: patient			
id	**person_id**	**room_id**	**ICD-11**
1	9	1	['CA01', 'BA00']
2	10	1	['BD11', '2F7A']
3	11	3	['EA80']
4	13	2	['EA80', 'BA00']
5	12	4	['BD11', 'EA80']
6	15	4	['2F7A', 'BA00', 'SA10']
7	16	2	['SA10']
8	14	4	['BA00', 'EA80', 'SA11']

Construction Step 4

Transformed tables:

Rooms

↓

Persons

↓

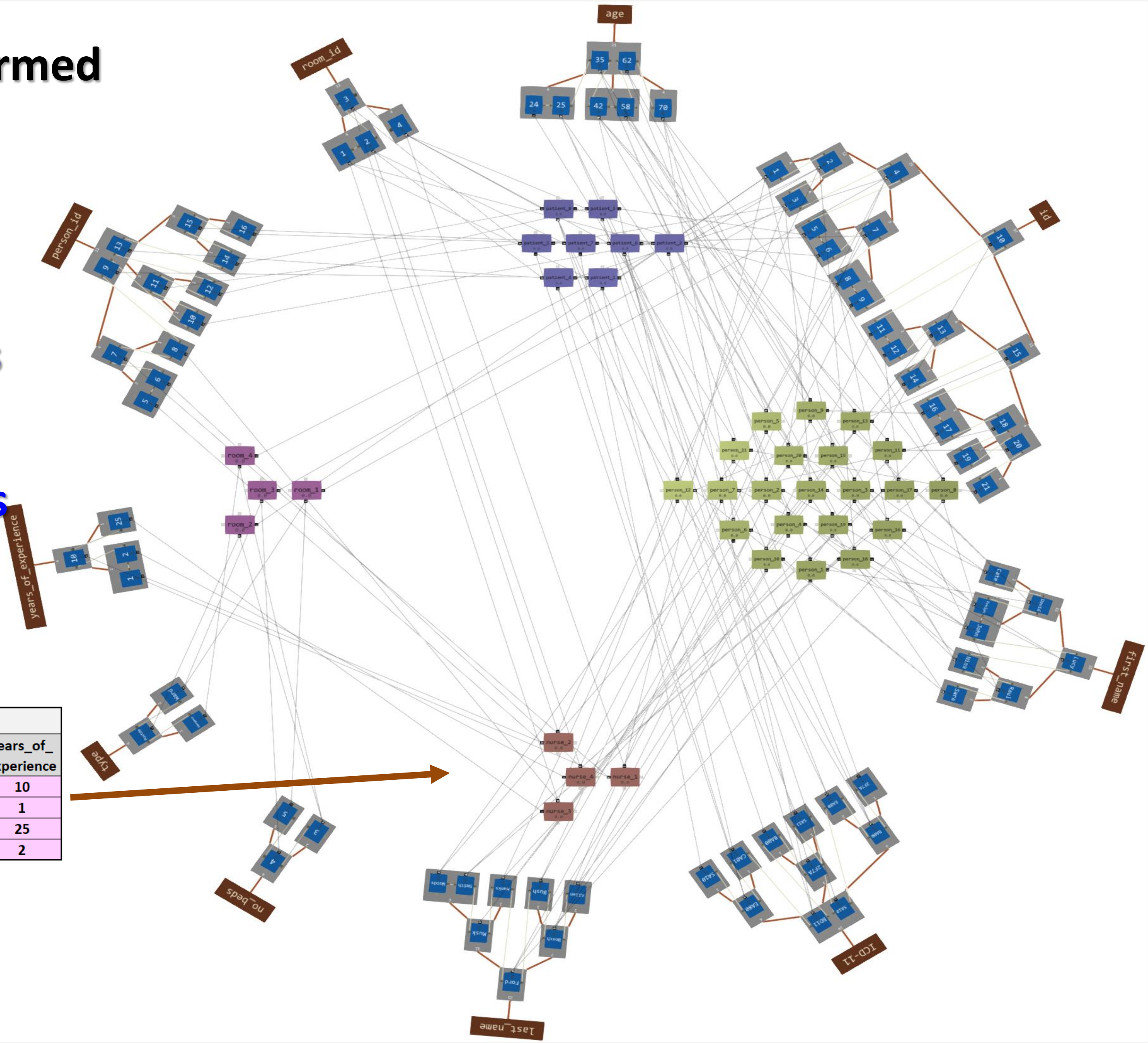
Patients

↓

Nurses

table: **nurse**

id	person_id	room_id	years_of_experience
1	7	4	10
2	5	3	1
3	8	1	25
4	6	2	2



Construction Step 5

Transformed tables:

Rooms

↓

Persons

↓

Patients

↓

Nurses

↓

Doctors

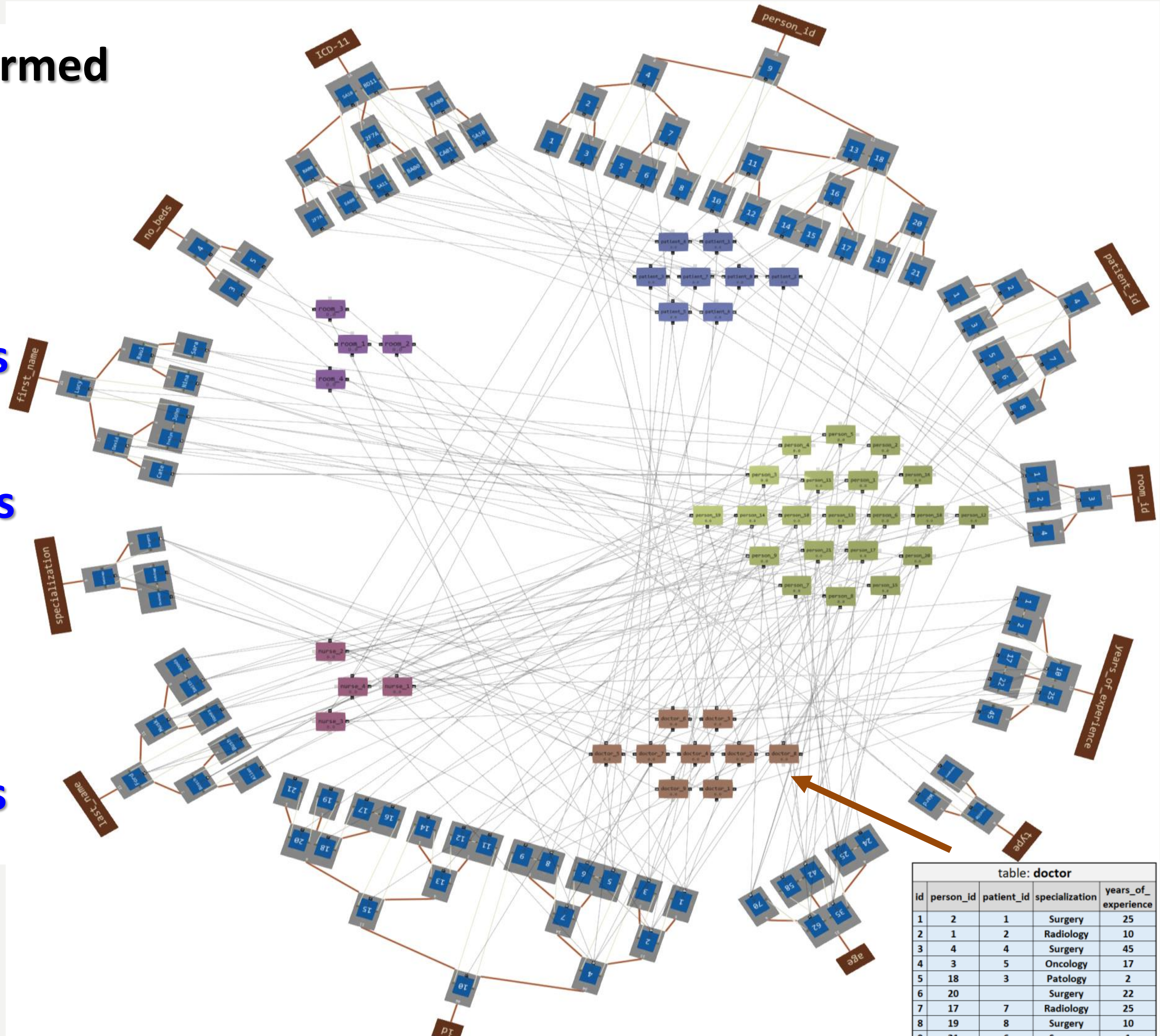
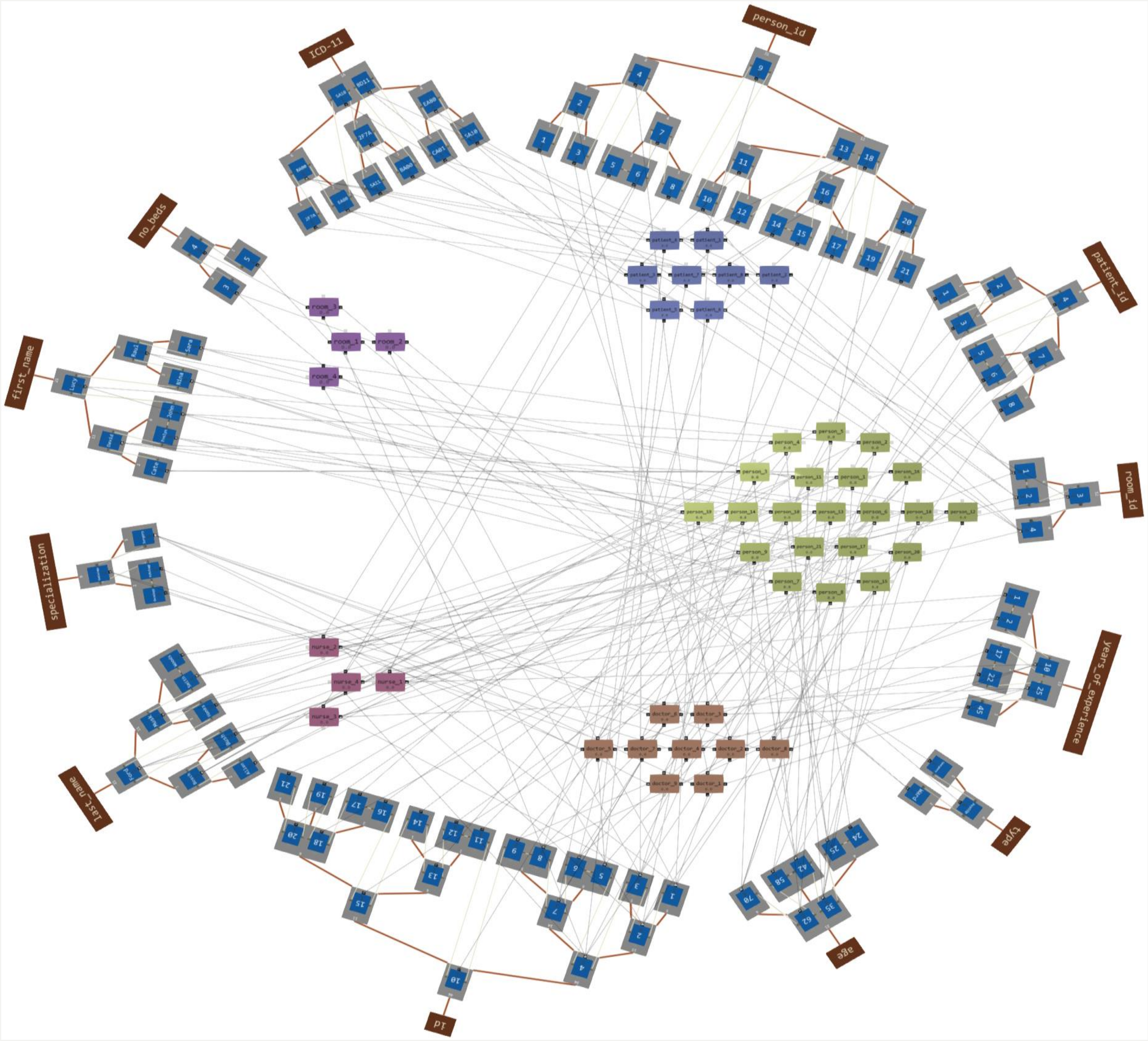


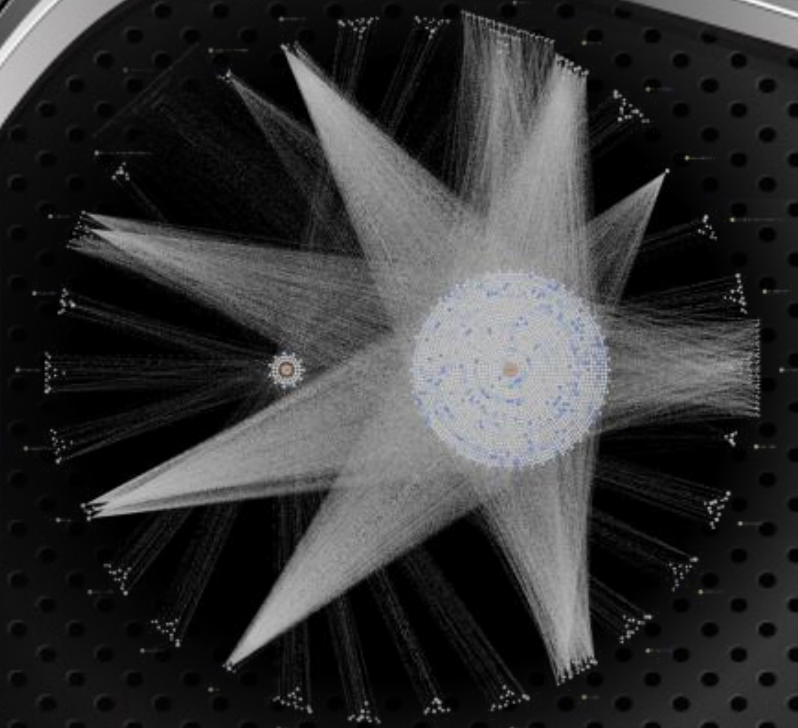
table: **doctor**

id	person_id	patient_id	specialization	years_of_experience
1	2	1	Surgery	25
2	1	2	Radiology	10
3	4	4	Surgery	45
4	3	5	Oncology	17
5	18	3	Patology	2
6	20		Surgery	22
7	17	7	Radiology	25
8	19	8	Surgery	10
9	21	6	Surgery	1

MAGN Constructed and Weights Calculated



ECML PKDD 2023



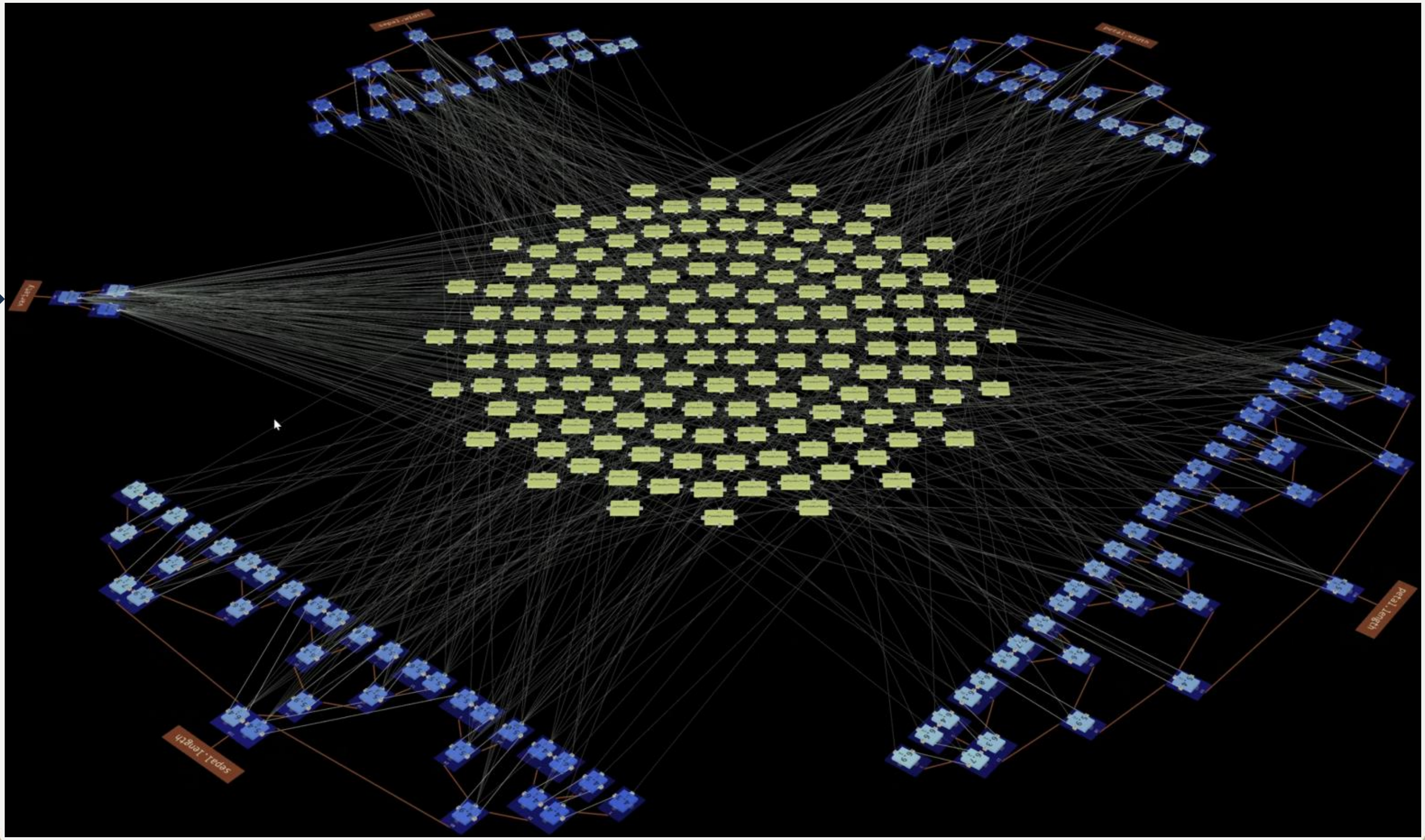
Multi-Associative Graph Networks for Classification and Regression

Construction and Training of Multi-Associative Graph Networks

MAGN Classification and Regression

MAGNs can be successfully used for classification and regression tasks defined by tables of training examples or databases, where we do not need to specify which attribute contains labels (classes, predicted values) before constructing MAGNs. We can do it at any time later!

ASA graph of class labels

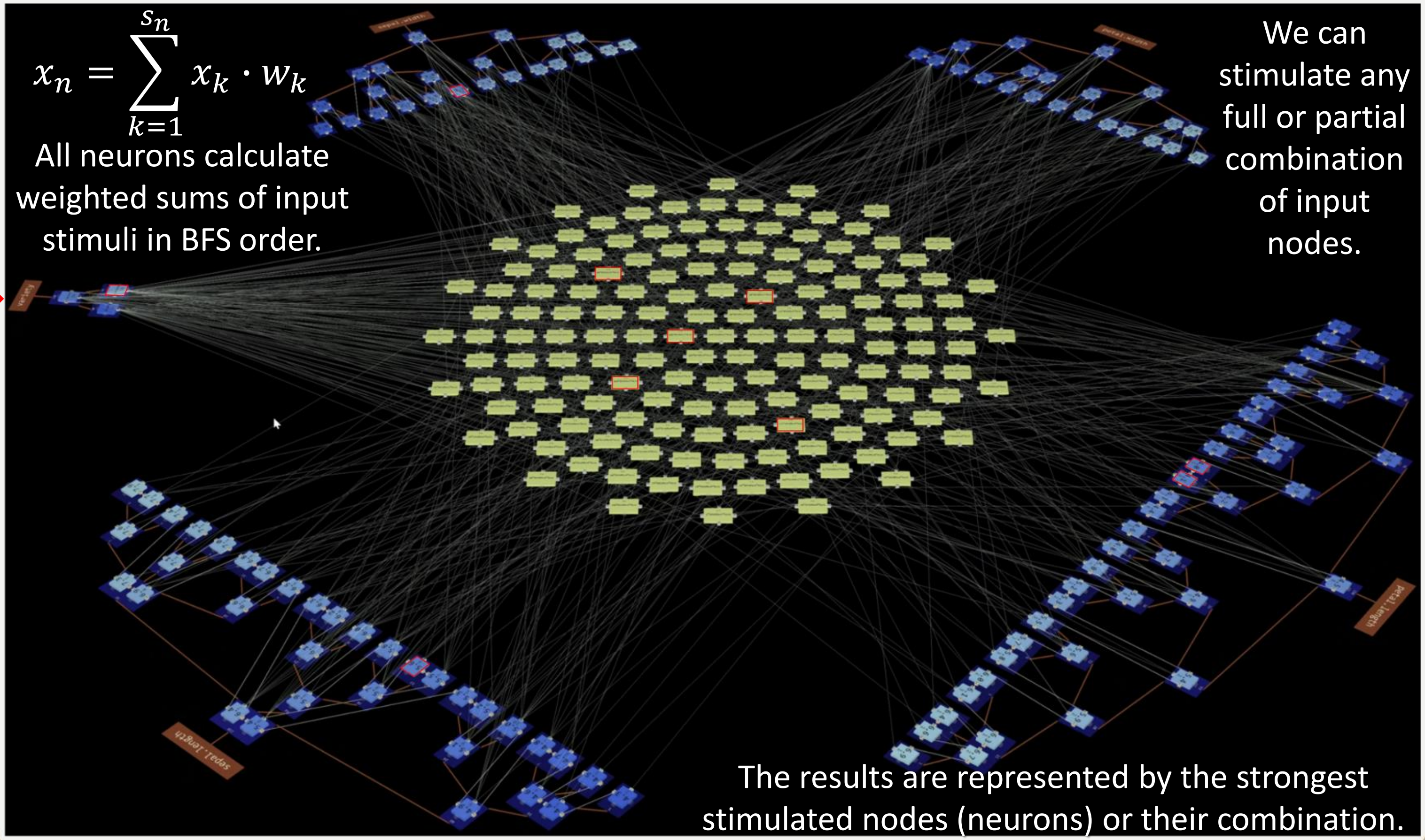


The MAGN constructed for IRIS training data.

Constructed MAGN Exploitation

MAGNs can be exploited in many different ways. We can freely choose input and output (target) attributes, stimulate any combination of input nodes, propagate stimulation through this network according to the calculated weights in the BFS order, and read out the mostly stimulated output nodes.

The strongest stimulated class neuron (node)



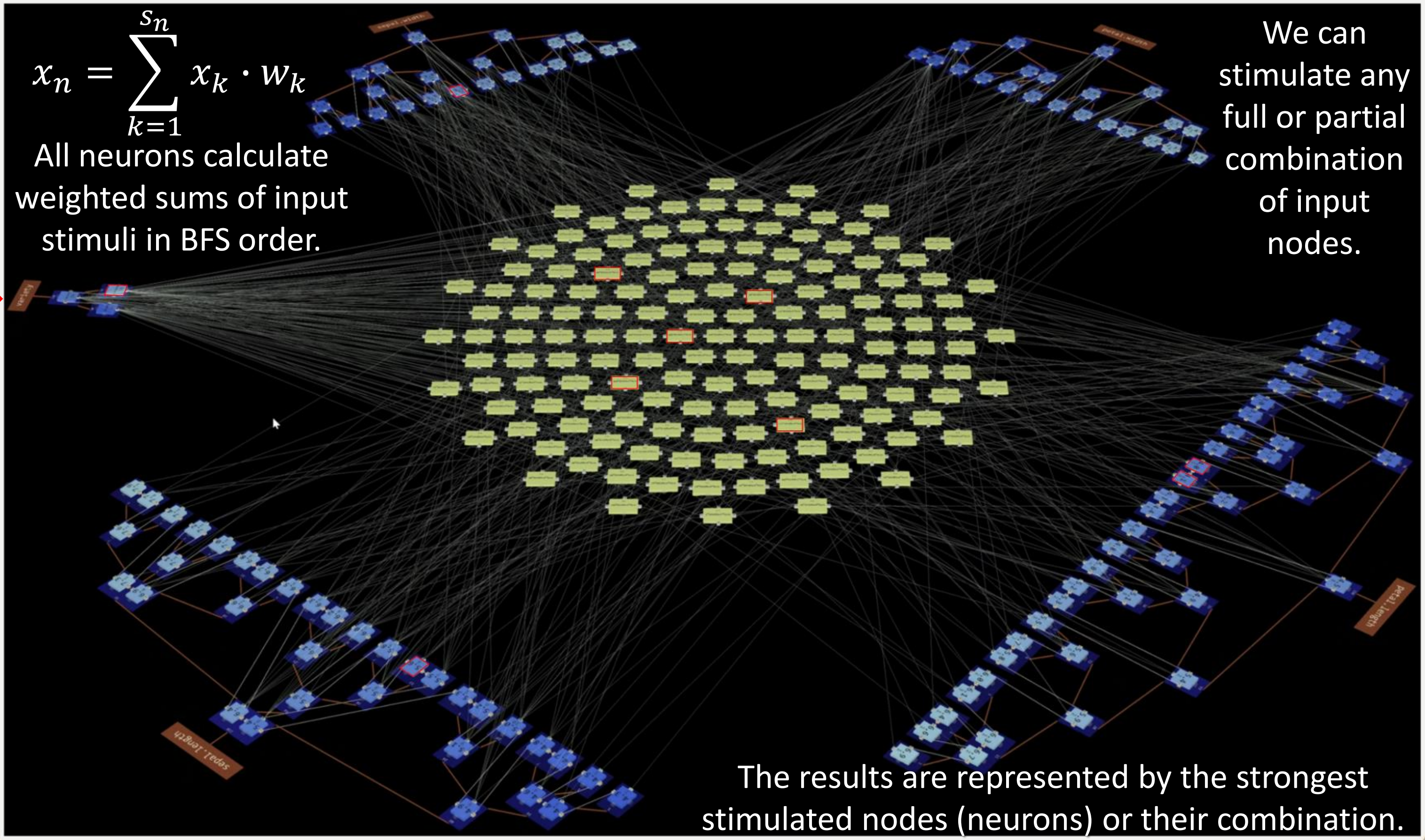
It points the strongest associated nodes to the input context.

Sparse MAGN Connections

Sparse MAGN connections represent **essential relationships** between values and objects represented by this graph structure. The **weights** of these connections reproduce the **importance of these relationships**.

Sparse connections resemble **hard attention**, while **weights soft attention**.

The strongest stimulated class neuron (node) represent the answer.



It points the strongest associated nodes to the input context.

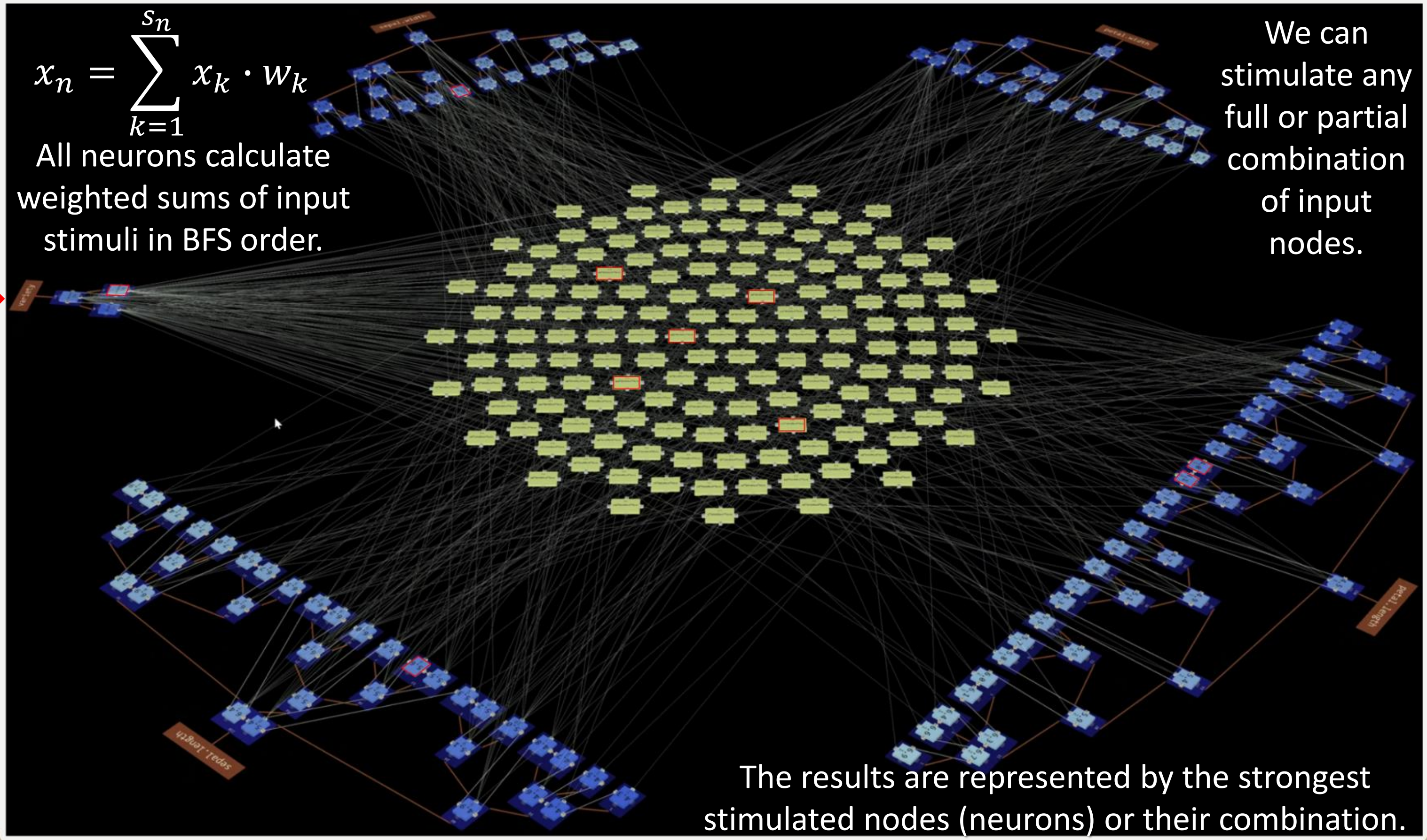
MAGN Classification & Regression

Classification results are pointed by **the strongest stimulated label nodes**.

Regression results are calculated based on **the strongest stimulated numerical nodes** of the attributes pointed as outputs (targets).

We can change targets without retraining the network and solve diverse tasks.

The strongest stimulated class neuron (node) represent the answer.



It points the strongest associated nodes to the input context.

ECML PKDD 2023

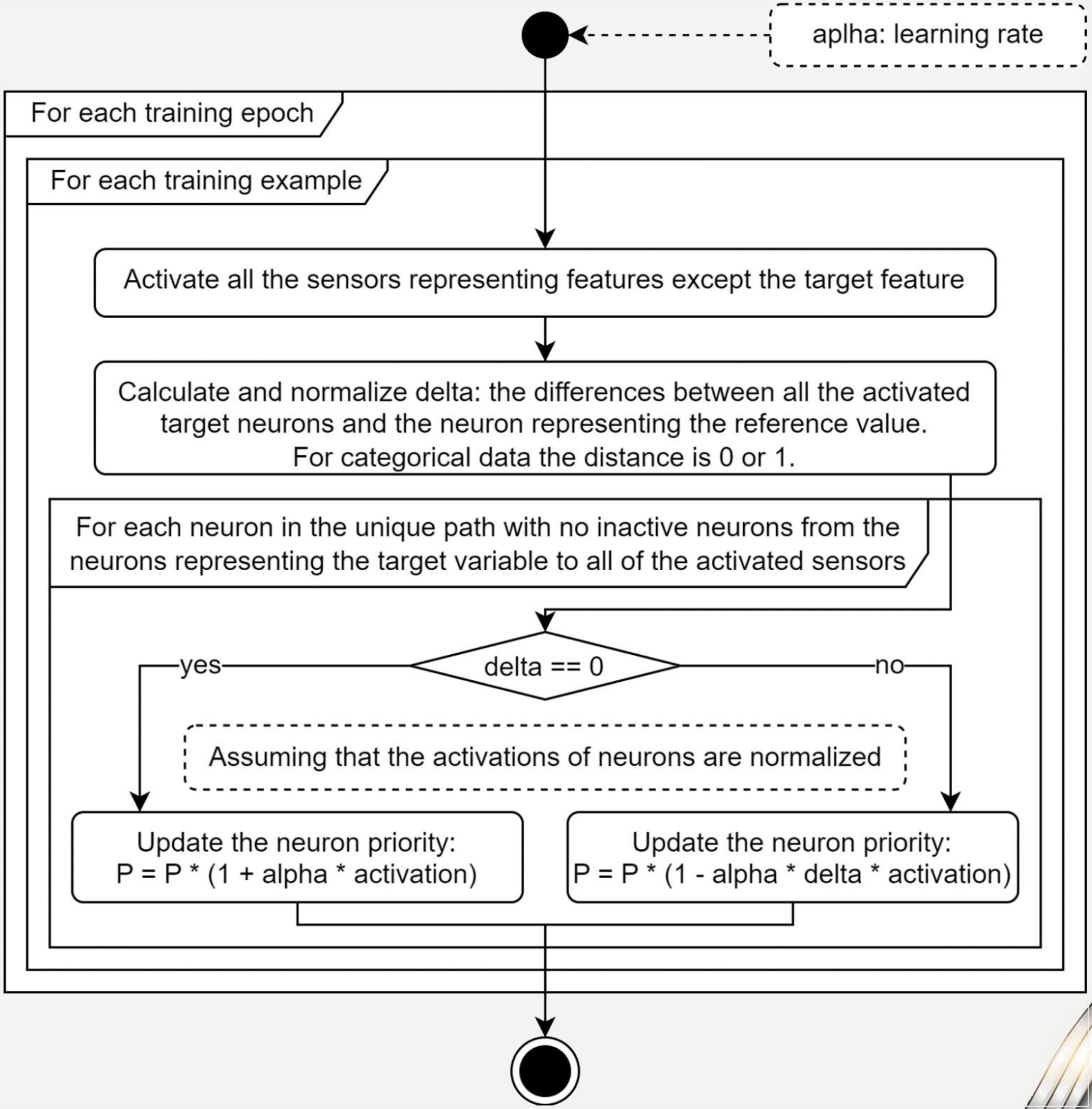
Prioritization Algorithm

for data and relationship soft-attention

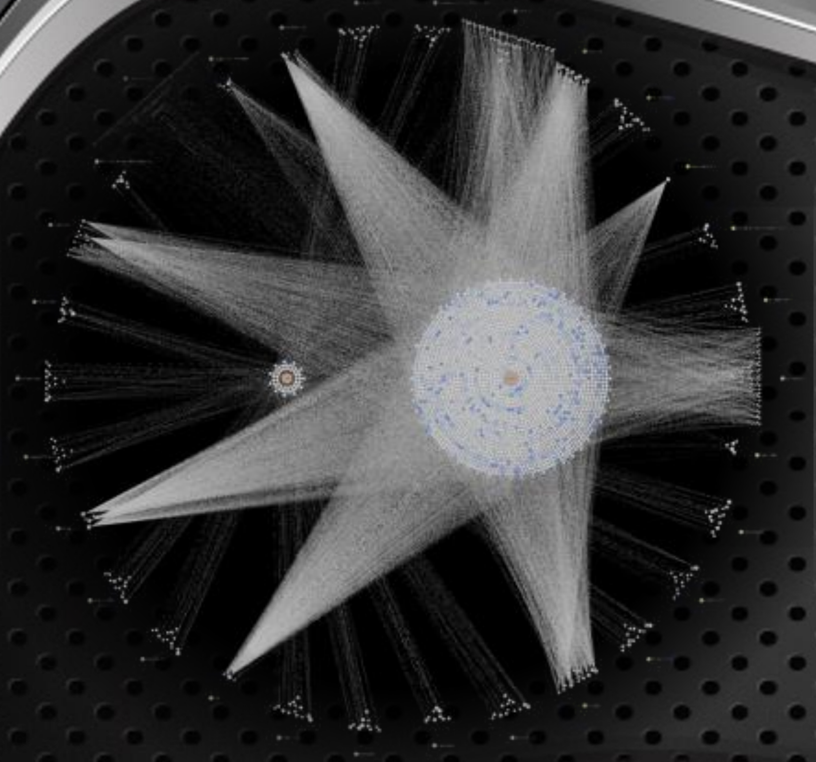
Construction and Training of
Multi-Associative Graph Networks

MAGN Prioritization Algorithm

It allows us to add attention to data and relationships, strengthening or weakening impact of the nodes of this structure to achieve still better results of regression and classification.



ECML PKDD 2023



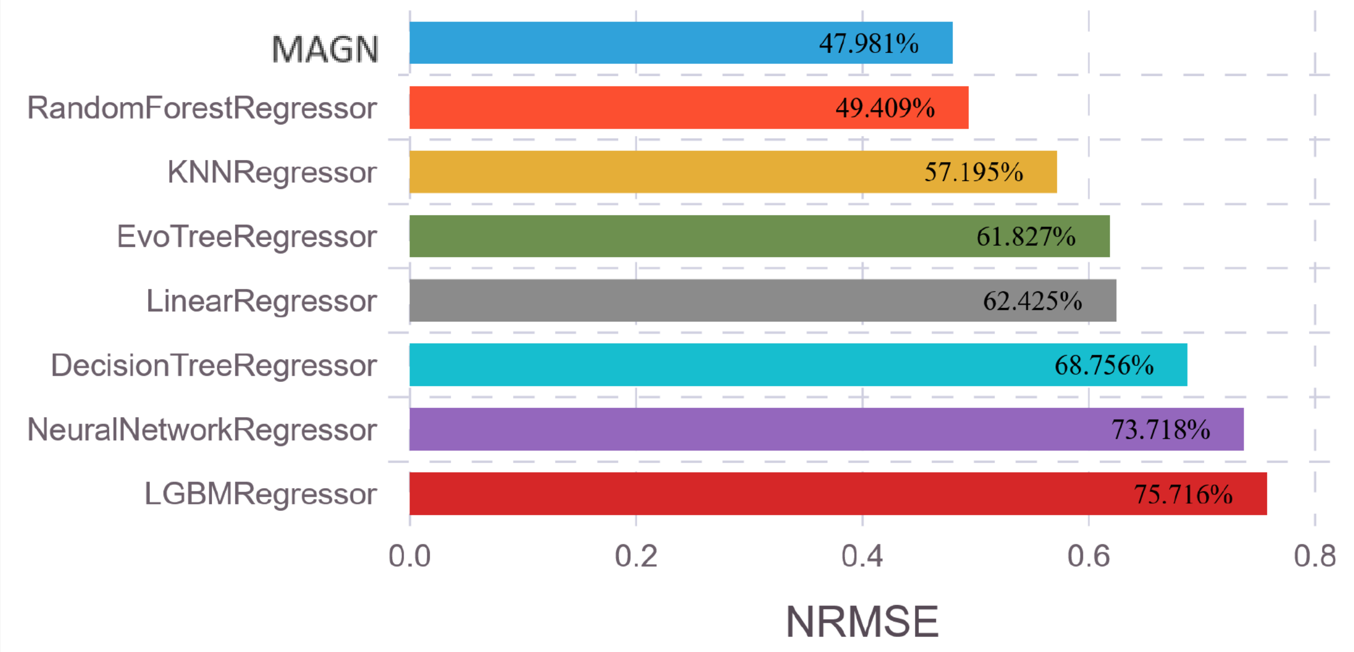
Multi-Associative Graph Networks

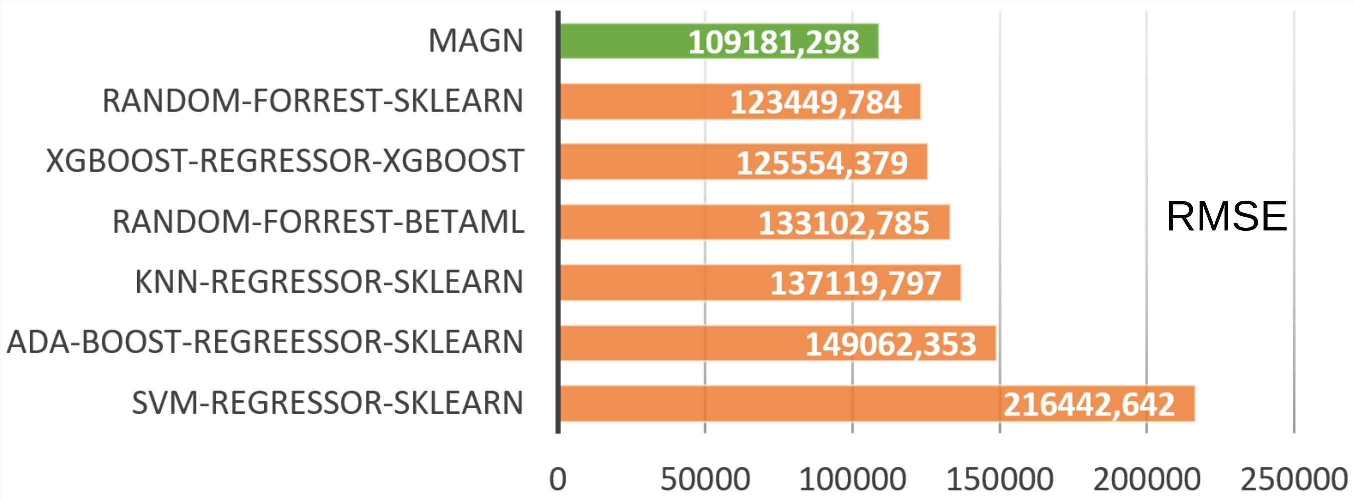
Experimental Results and Comparisons to SOTA Methods

Construction and Training of Multi-Associative Graph Networks

Experiments & Comparisons

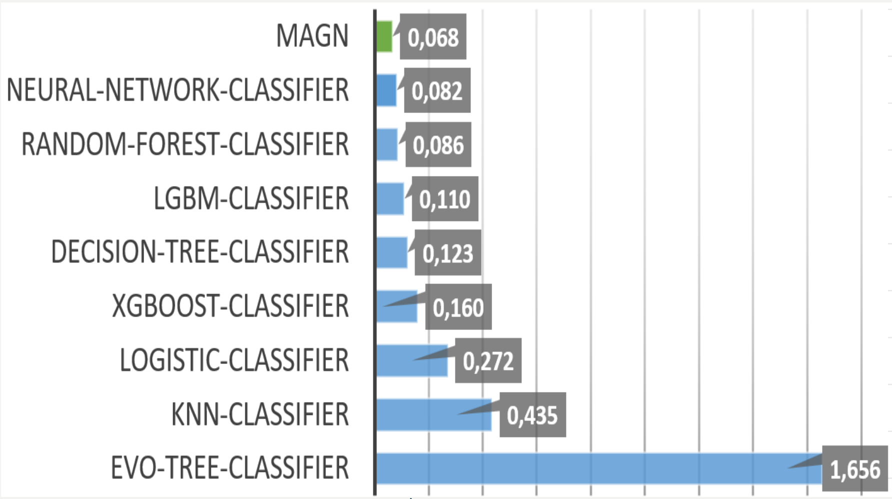
Comparisons of the results collected for regression 18 training datasets:

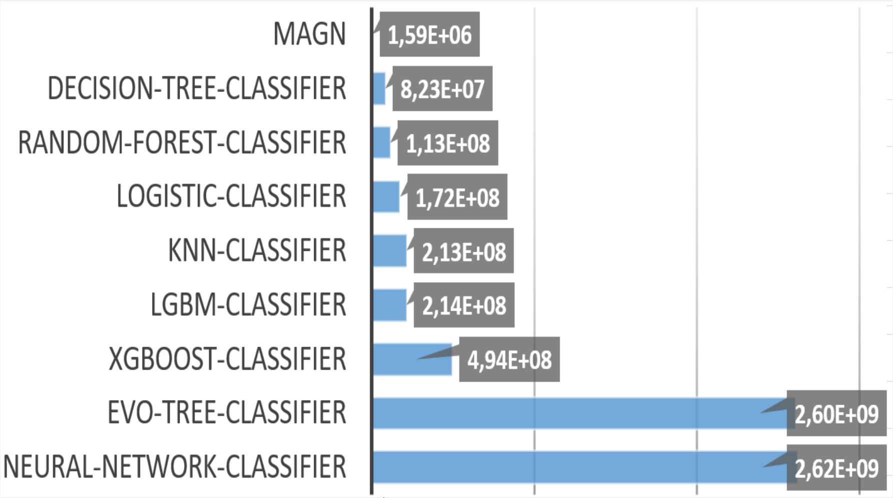




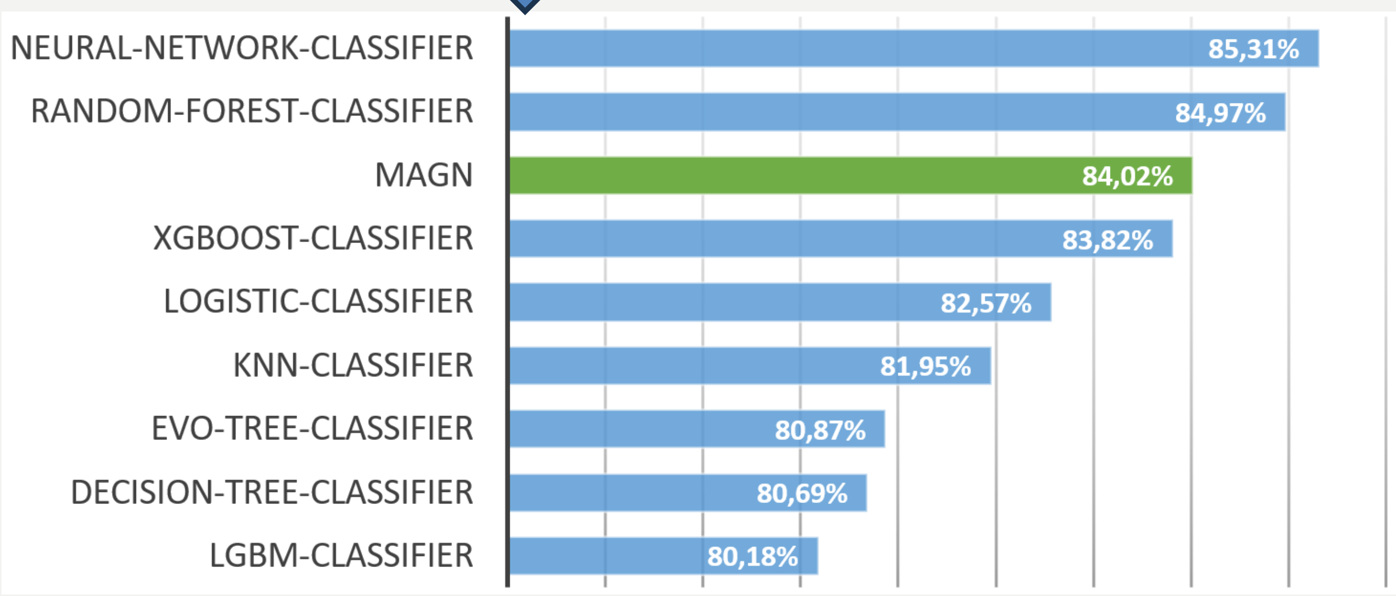
Experiments & Comparisons

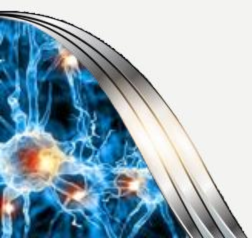
Comparisons of the results collected for classification 73 training datasets:

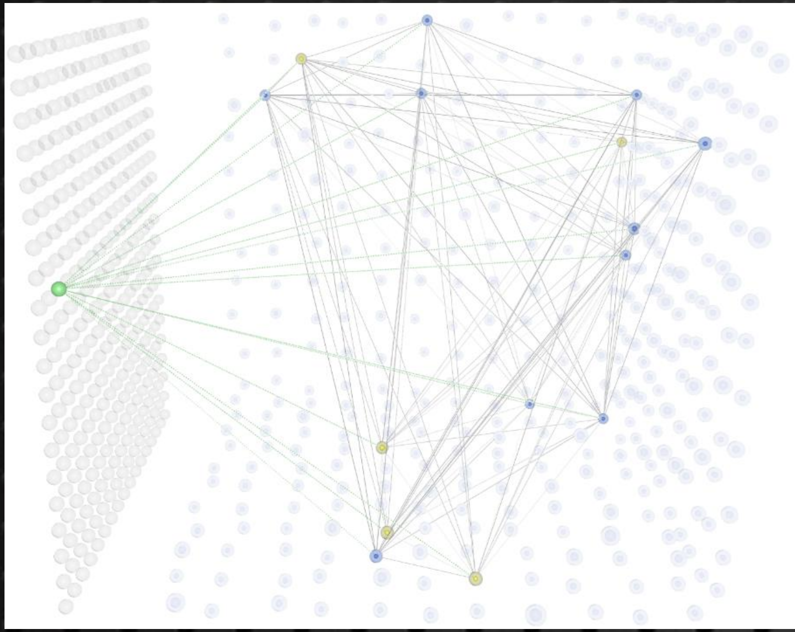




Execution Time ⇧ ⇩ **Mean Accuracy** ⇧ **Allocated Memory**







Commercial Deployment & Conclusions

MAGN implementation and source code

The MAGN source code is available at
https://github.com/danbulnet/witchnet

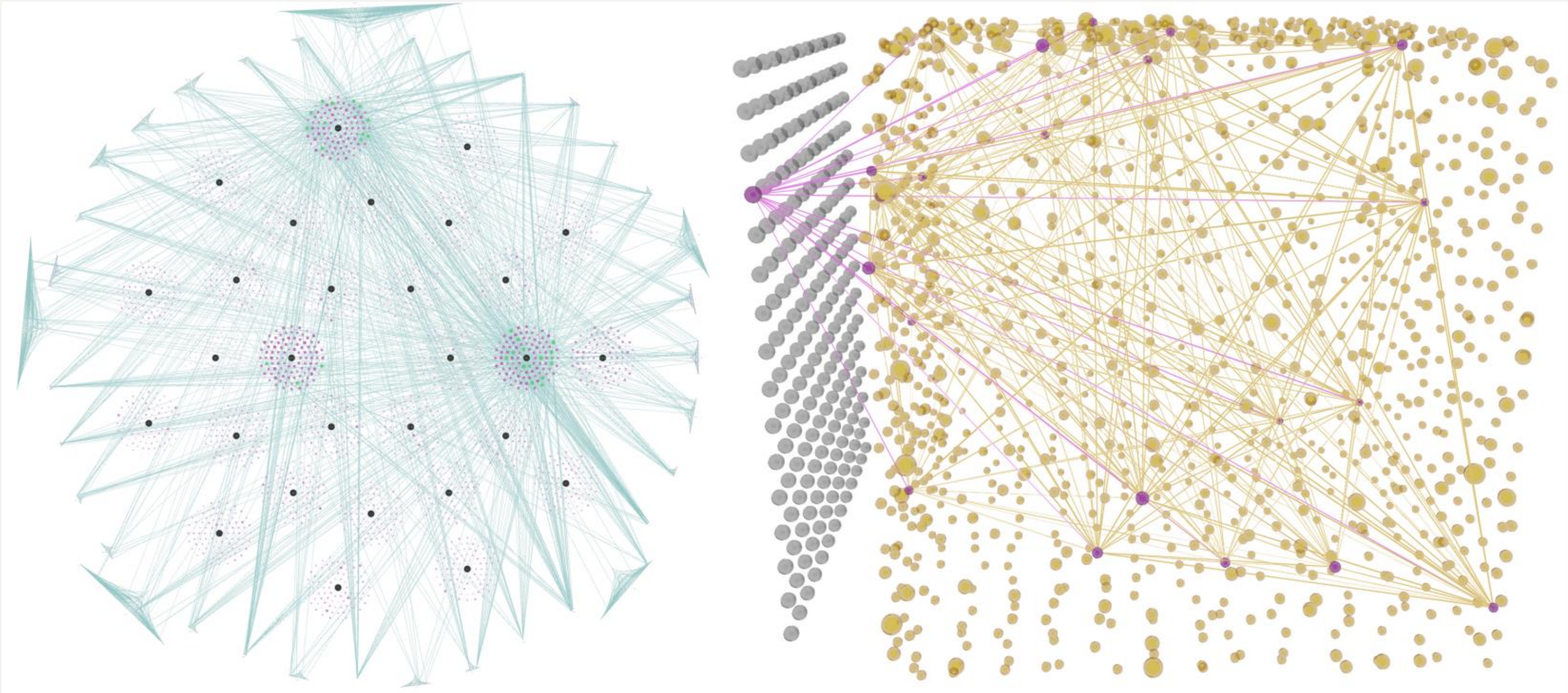
The source code can be downloaded and compiled on multiple platforms. There are no specific minimal hardware requirements; however, since this is an in-memory model, the amount of RAM should scale with the amount of data being modeled. The code was tested on popular 64-bit operating systems, such as Windows 11, MacOS 13, and Linux Endevaour OS 2023.

Benchmark source code is available at
https://github.com/danbulnet/WitchnetBenchmarks.jl

Commercial Deployment

MAGNs are under commercial deployment in cooperation with:

grape up®

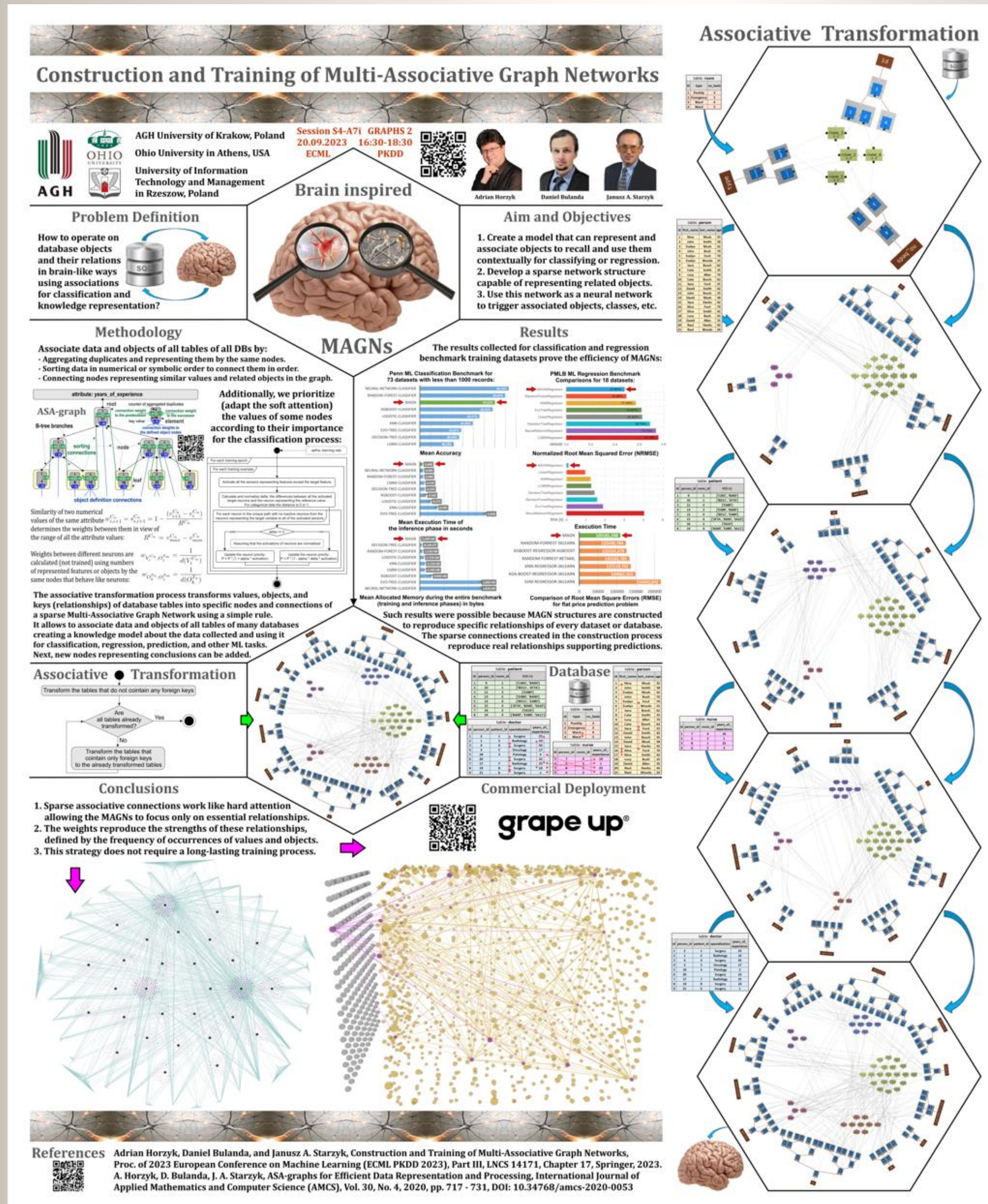


https://grapeup.com/blog/associative-knowledge-graphs/

Conclusions and Final Remarks

- ✓ **MAGNs** are fully **scalable** and **explainable** models that can be used for **classification** or **regression** of vectorized data stored in databases.
- ✓ They can be very **quickly** constructed and **fast** evaluate and answer.
- ✓ Training data can be **updated** at any time **without model „retraining"**.
- ✓ They can be **constructed without specifying „labels"** that can be chosen later, i.e., every attribute can be chosen as target without „retraining".
- ✓ They can be used as associative knowledge graphs to store information about associated objects of one or many databases **parsimoniously**.
- ✓ They can be used for **searching** for various data and relationships in any given context. The results are **legible** and **easy interpretable**.
- ✓ They can **cluster, classify, predict, recommend, group, recognize, find associations**... of any vectorized data stored in RDBs used in **MAGNs**.

Welcome to the poster for further discussions



Research papers:

Construction and Training of Multi-Associative Graph Networks

Adrian Horzyk

Daniel Bulanda

Janusz A. Starzyk

AGH University of Krakow - University of Information Technology and Management in Rzeszow - Ohio University in Athens

Bibliography and Associated Works

- ✓ A. Horzyk, J.A., Starzyk, J. Graham, Integration of semantic and episodic memories, IEEE transactions on neural networks and learning systems 28(12), 2017, 3084–3095.
- ✓ Basawaraj, J.A. Starzyk, A. Horzyk, Episodic memory in minicolumn associative knowledge graphs. IEEE transactions on neural networks and learning systems 30(11), 2019, 3505–3516.
- ✓ A. Horzyk, Associative graph data structures with an efficient access via AVB+trees. In: 2018 11th International Conference on Human System Interaction (HSI), IEEE, 2018, 169–175.
- ✓ A. Horzyk, Associative representation and processing of databases using DASNG and AVB+trees for efficient data access. In: Knowledge Discovery, Knowledge Engineering and Knowledge Management: 9th International Joint Conference, IC3K, 2017, Funchal, Madeira, Portugal, November 1-3, 2017, Revised Selected Papers 9, Springer, 2019, 242–267.
- ✓ A. Horzyk, D. Bulanda, J.A. Starzyk, ASA-graphs for efficient data representation and processing. International Journal of Applied Mathematics and Computer Science 30(4), 2020, 717–731.
- ✓ https://grapeup.com/blog/associative-knowledge-graphs/
- ✓ https://github.com/PrzemyslawStok/Structural-Properties-of-Associative-Knowledge-Graphs