Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

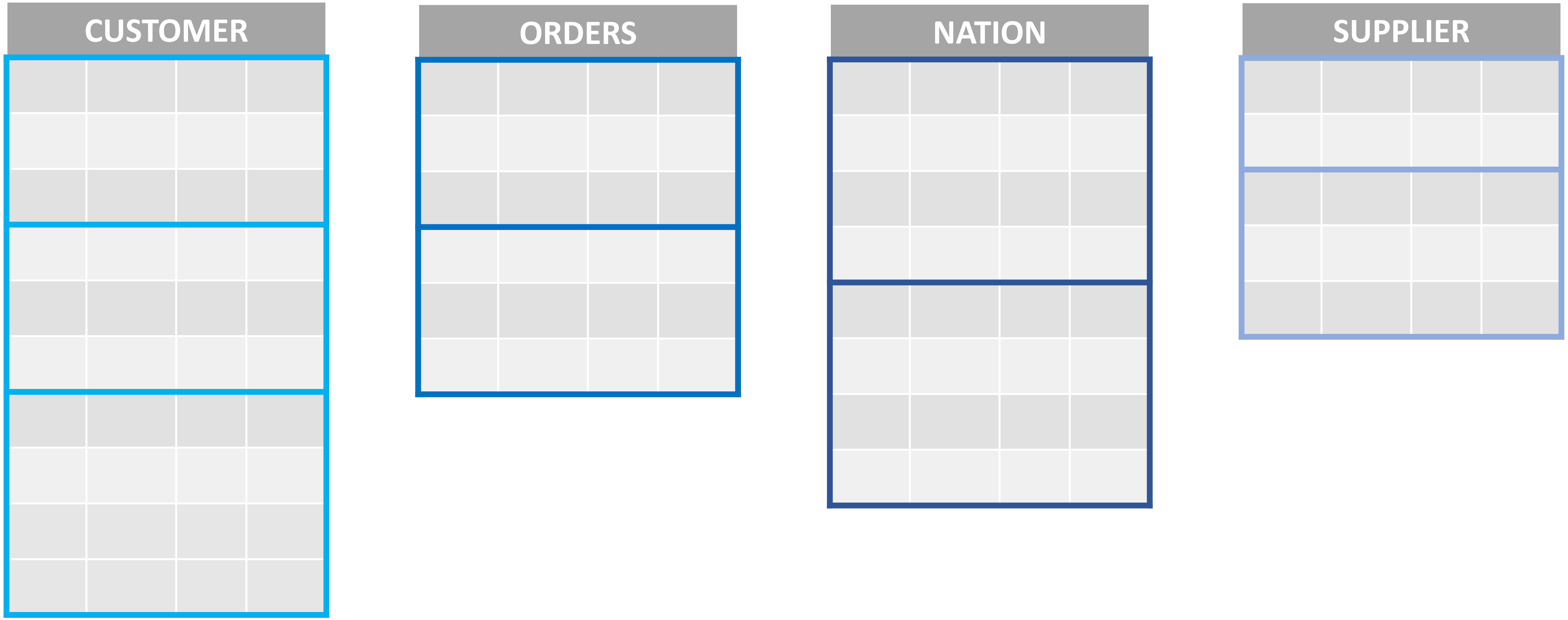
DBIS LAB
RPTU Kaiserslautern-Landau

Tuple Bubbles: Learned Tuple Representations for Tunable Approximate Query Processing

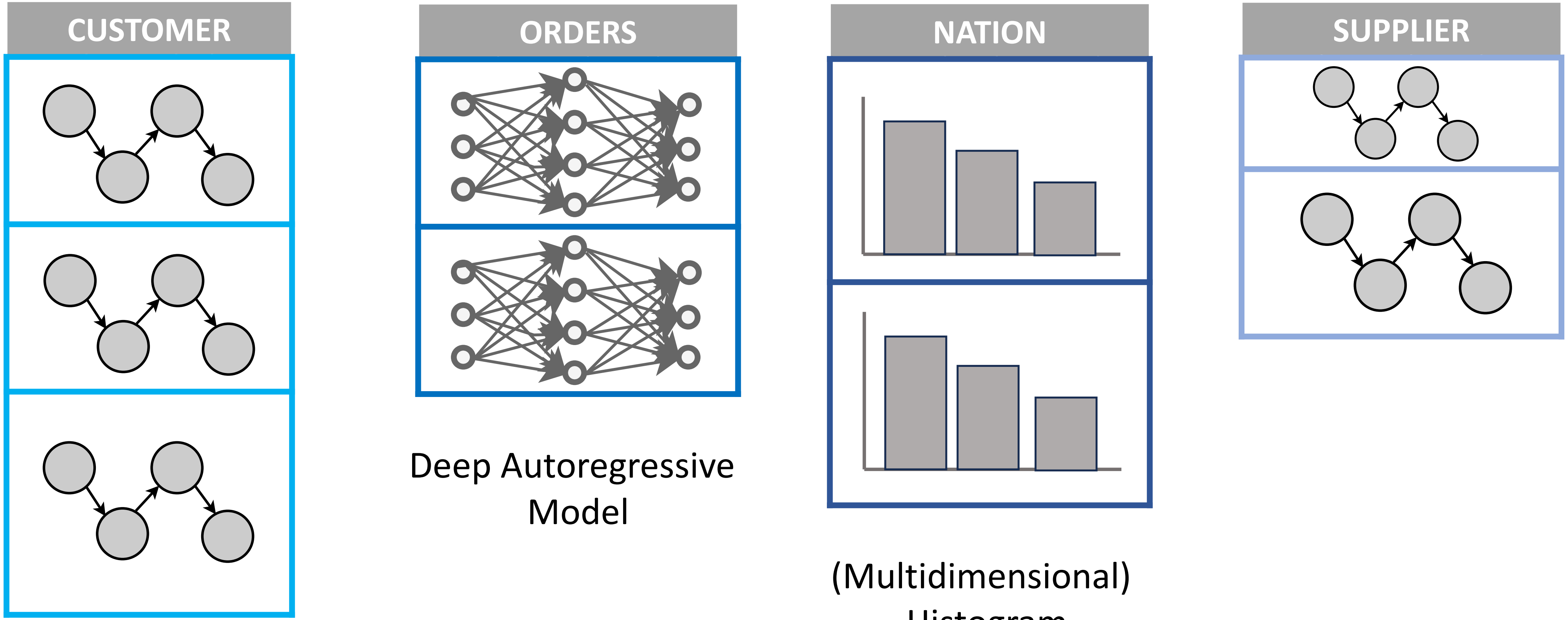
Damjan Gjurovski
damjan.gjurovski@cs.rptu.de
RPTU Kaiserslautern-Landau

Sebastian Michel
sebastian.michel@cs.rptu.de
RPTU Kaiserslautern-Landau

Group Tuples into Tuple Bubbles



Create Bubble Summaries



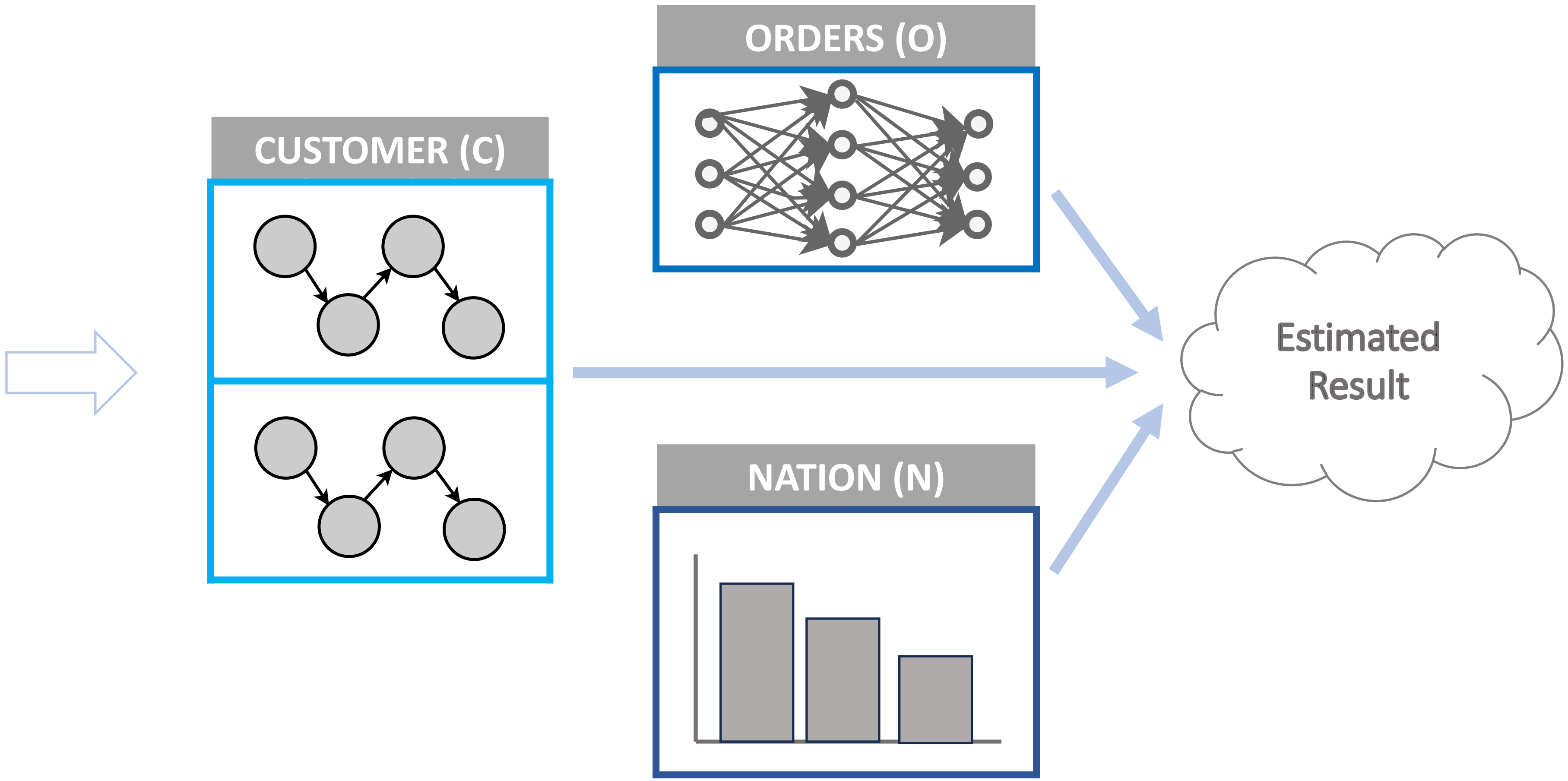
Bayesian Network

Deep Autoregressive Model

(Multidimensional) Histogram

Estimate Query Results from Summaries

```
SELECT COUNT(*)
FROM N, C, O
WHERE C.BAL > 10
AND O.CLERK = 'C#1'
AND N.N_PK = C.N_PK
AND C.C_PK = O.C_PK
```



Benefits

- Computation performed over less data
- Faster execution
- Less memory
- Applicable in a distributed environment

Three Core Tasks

1. Organize tuples into Bubbles
2. Create representative but compact bubble summaries
3. Query processing over bubble summaries

Creating Tuple Bubbles

- Organize data from tables into groups:
  - Horizontal partitioning (e.g. primary key)
  - Identify dependencies between tables
  - Similarity-based
- Optimization based on foreign key relationships
- Affects selection of bubbles, estimation accuracy, and execution time

o_key	c_key	price	date	
1	3	22.3	01.03.22	TBO_1
2	4	75.0	01.03.22	TBO_1
3	4	444.0	02.03.22	TBO_1
4	4	399.6	02.03.22	TBO_2
5	2	400.5	03.03.22	TBO_2

c_key	name	
2	C2	TBC_1
3	C3	TBC_1
4	C4	TBC_1

Three Core Tasks

1. Organize tuples into Bubbles
2. Create representative but compact bubble summaries
3. Query processing over bubble summaries

Bubble Summaries

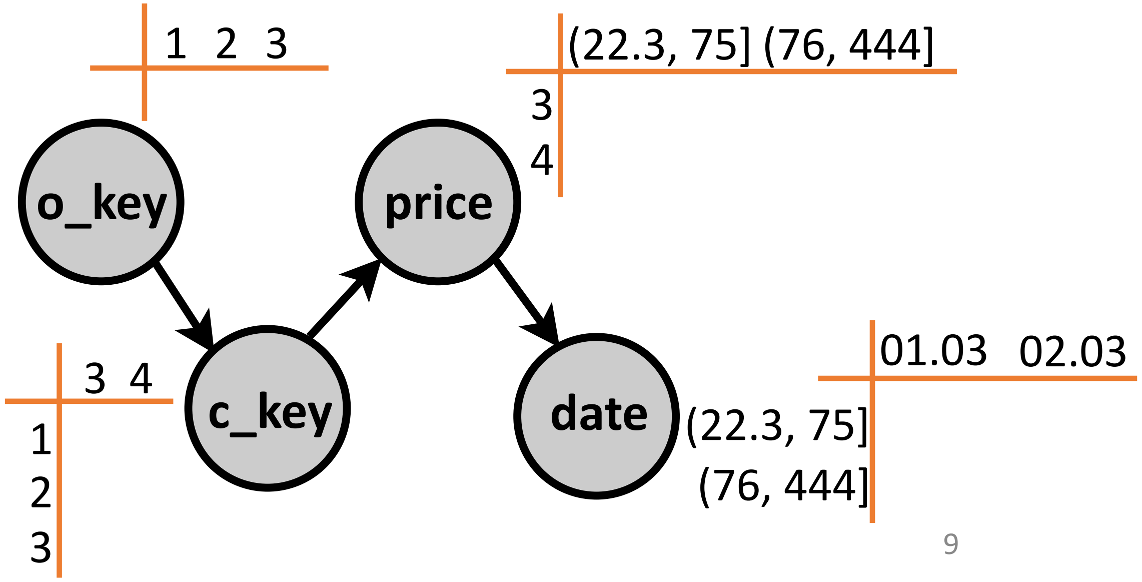
- Create representative but compact models of the bubbles
- One summary per bubble
- Considered summarization models:
  - Bayesian networks
  - Deep autoregressive models

Summaries as Bayesian Networks

- Chow-Liu tree structure learning algorithm
- Probability distribution per node given single parent
- $card(A_i)^{p+1}$ values per attribute
- Keep k most appearing values and group remaining values into b buckets
- Include primary and foreign keys

o_key	c_key	price	date
1	3	22.3	01.03.22
2	4	75.0	01.03.22
3	4	444.0	02.03.22

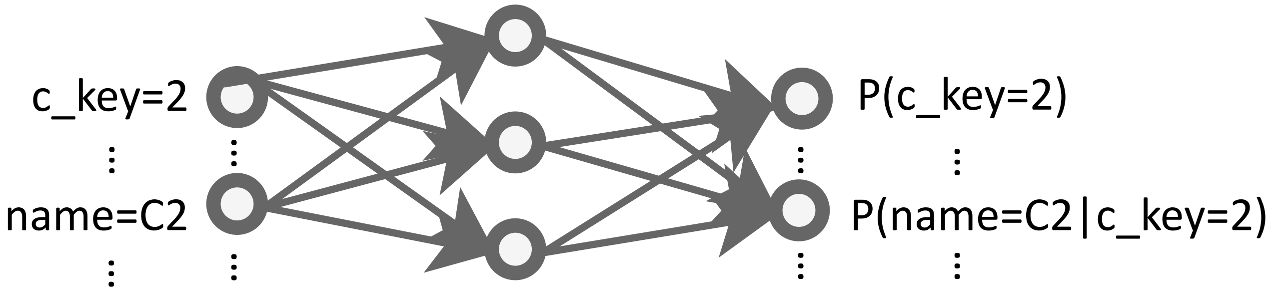
$\Big\}$ TBO_1



Summaries as Deep Autoregressive Models

- Learning over the data
- Estimates density per value conditioned on previous attributes
- Per-attribute losless compression
- Requires complete join result for training

c_key	name
2	C2
3	C3
4	C4



Three Core Tasks

1. Organize tuples into Bubbles
2. Create representative but compact bubble summaries
3. Query processing over bubble summaries

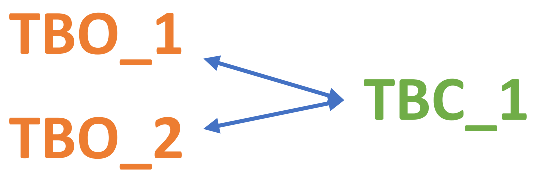
Query Processing

- Access all bubbles based on query relations
- Combination of bubbles for the query relations
- Estimation over single bubbles:
  - Variable elimination (Bayesian networks)

$$P(p) = \sum_o P(o) \sum_c P(c|o) * P(p|c) \sum_d P(d|p)$$

  - Progressive sampling (Bayesian networks and deep autoregressive models)

$$P_o(q) = \prod_{i=1}^{4} P(A_i|A_{<i})$$



Estimating Aggregate Queries

- Variable elimination
  - Final probabilities as $P(A_i|E)$
  - Consider ranges for binned attributes
- Progressive sampling
  - Ordering of attributes affects final probabilities
    - Aggregation attribute can be before query predicates
  - Generate samples for aggregation attribute as query predicates
  - Ranges as samples

```
SELECT SUM(c_key) FROM orders
WHERE price > 75
```

o_key	c_key	price	date
1	2	3	4

Join Queries with Bayesian Networks

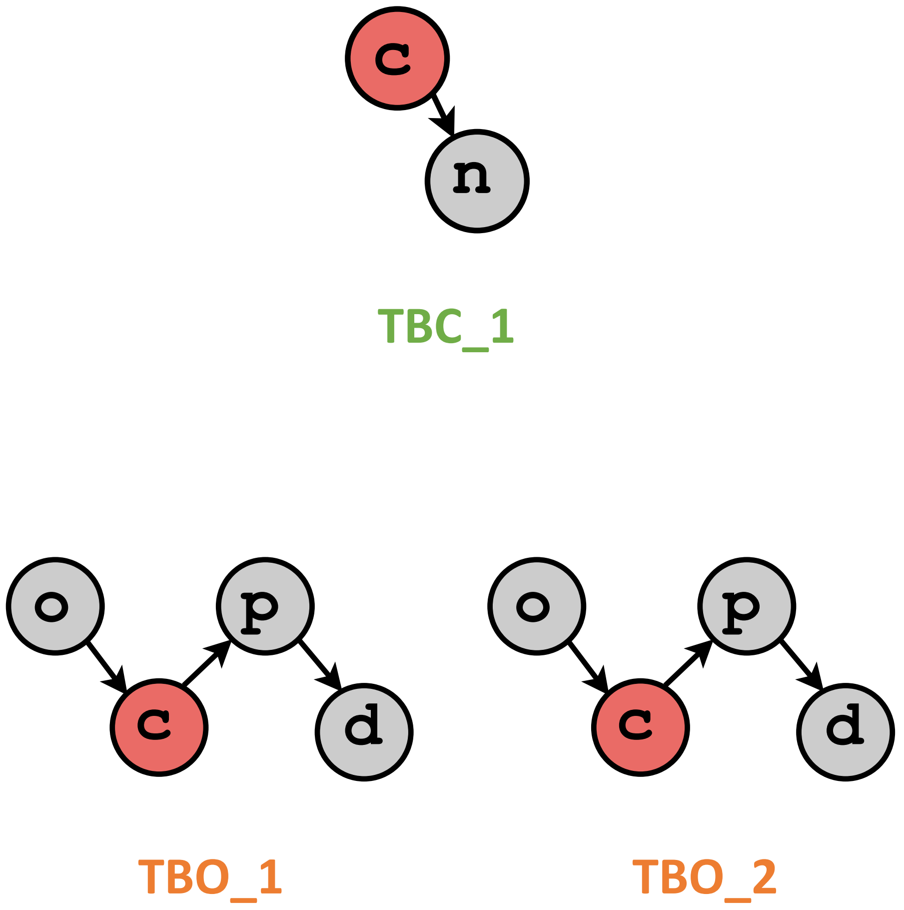
```
SELECT SUM(price)
FROM customer c, orders o
WHERE c.c_key = o.c_key
AND c.name = 4
AND o.date > 02.03.22
```

ORDERS

o_key (o)	c_key (c)	price (p)	date (d)

CUSTOMER

c_key (c)	name (n)



Join Queries with Bayesian Networks

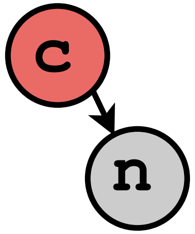
```
SELECT SUM(price)
FROM customer c, orders o
WHERE c.c_key = o.c_key
AND c.name = 4
AND o.date > 02.03.22
```

ORDERS

o_key (o)	c_key (c)	price (p)	date (d)

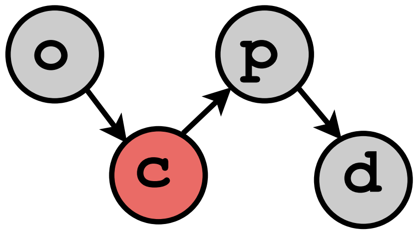
CUSTOMER

c_key (c)	name (n)

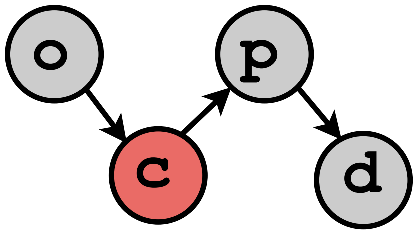


TBC_1

1



TBO_1



TBO_2

Join Queries with Bayesian Networks

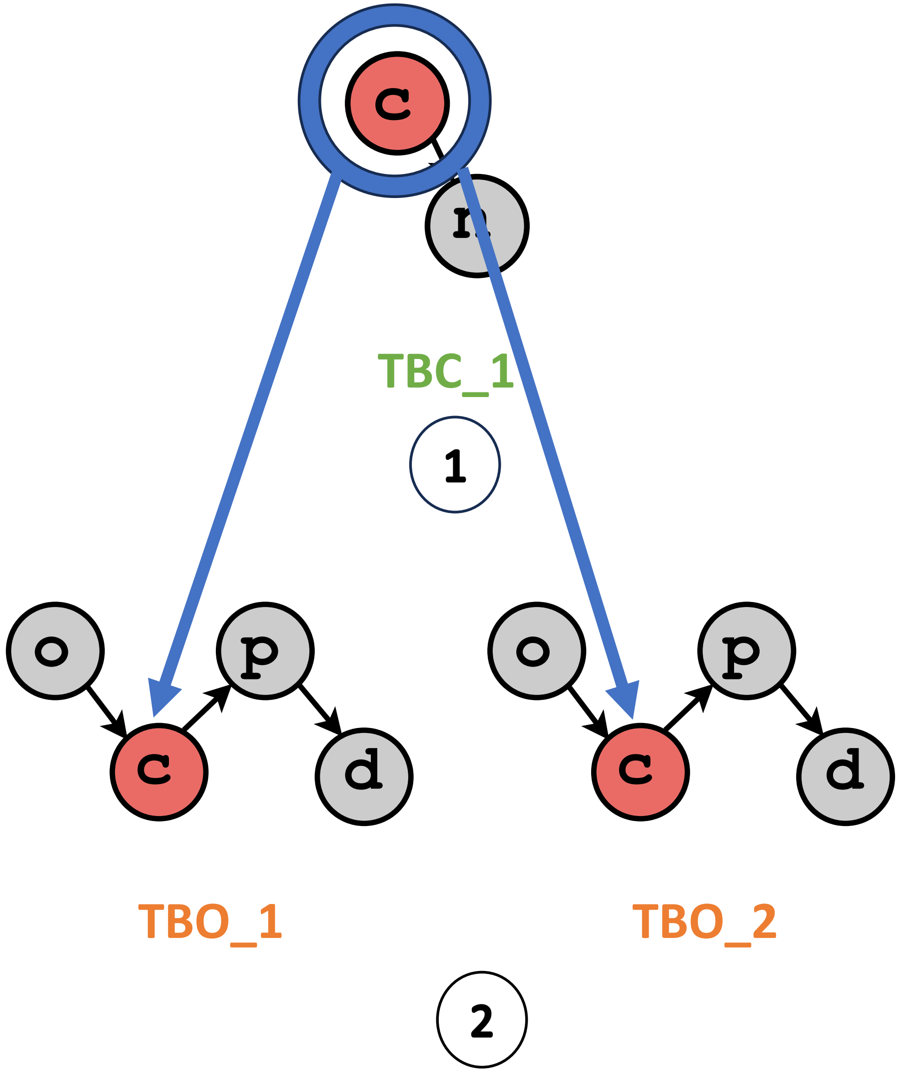
```
SELECT SUM(price)
FROM customer c, orders o
WHERE c.c_key = o.c_key
AND c.name = 4
AND o.date > 02.03.22
```

ORDERS

o_key (o)	c_key (c)	price (p)	date (d)

CUSTOMER

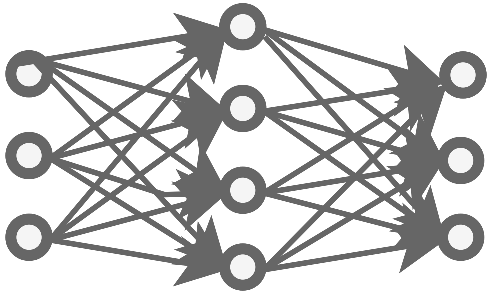
c_key (c)	name (n)



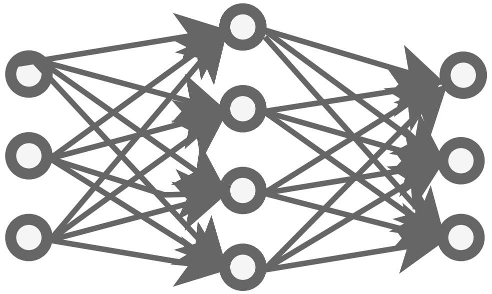
Join Queries with Autoregressive Models

- Identify relevant summaries with respect to the query
- Training performed over the complete join
- Number of models is the number of different joinable bubbles

```
SELECT SUM(price)
FROM customer c, orders o
WHERE c.c_key = o.c_key
AND c.name = 4
AND o.date > 02.03.22
```



TBO_1 and TBC_1



TBO_2 and TBC_1

Experimental Evaluation

- Setup
  - NVidia GeForce RTX 2080 Ti GPU (Autoregressive model)
  - Intel Xeon E5-2603 v4 CPU@1.7GHz, 128 GB RAM (Bayesian network; Competitors)
- Datasets
  - TPC-H, IMDB, Intel Wireless
- Competitors
  - PostgreSQL 14
  - VerdictDB (VDB) *Park et al. SIGMOD 2018*
  - Wander join (WJ) *Li et al. ACM Trans. Database Syst. 2019*
  - KD-Pass *Liang et al. SIGMOD 2021*
  - AQP++ *Peng et al. SIGMOD 2018*

Approach Variants

- Bayesian networks (TB_BN)
  - Horizontal partitioning (k) with i bubbles per relation (TB_BN_i)
  - Join on foreign key with one network per join (TB_BN_J)
  - Horizontal partitioning and join on foreign key (TB_BN_J_i)
- Deep autoregressive mode (TB_AR)
  - Horizontal partitioning (k) with i bubbles per join (TB_AR_i)

Execution Time and Memory (TPC-H)

- Considered 150 queries
  - 2-5 joins and 2-5 predicates
- Maximal number of partitions $\boldsymbol{k}$ is 3
- Partitioning threshold $\boldsymbol{\theta}$ is 500 000
- Number of buckets between 60 and 200
- Storing 40 to 100 most common values

	Avg. Time (PS/VE)	**Memory**
Approach	ms	MB
PostgreSQL	1124.9	1399
TB_BN	**12.01** / 58.7	**4.8**
TB_BN_1	**10.3** / 45	**4.7**
TB_BN_J	**16.4** /160.4	**10.8**
TB_BN_J_1	**16** / 150	**17.3**
TB_AR	41.3	36.9
TB_AR_1	40.1	36.6
TB_AR_2	40.01	73.2
TB_AR_3	40.01	109.8
VDB 10%	96.1	147.5
VDB 50%	874.1	359
WJ	143.3	702

Accuracy (TPC-H)

- Considered 150 queries
  - 2-5 joins and 2-5 predicates
- Number of partitions $\boldsymbol{k}$ is 3
- Partitioning threshold $\boldsymbol{\theta}$ is 500 000
- Buckets between 60 and 200
- 40 to 100 most common values
- $q_{error} = \max(\frac{true(q)}{est(q)}, \frac{est(q)}{true(q)})$

Approach	Q-error (PS/VE) median	Q-error (PS/VE) max	Q-error (PS/VE) avg
PostgreSQL	1.0	1.0	1.0
TB_BN	3.9 / 3.3	$5.3*10^4$ / $5.3*10^4$	1064.0 / 1049.4
TB_BN_1	2.3 / 2.29	$5.3*10^4$ / $3.8*10^4$	1101.9 / 872.1
TB_BN_J	1.2/ **1.018**	**9740.0 / 9740.0**	**220.1 / 241.5**
TB_BN_J_1	2.02 / 2.006	$3.1*10^4$ / $1.1*10^6$	861.4 / $1.1*10^4$
TB_AR	1.56	**9805**	**408.29**
TB_AR_2	1.58	$4.1*10^5$	4013.15
TB_AR_3	1.36	$4.1*10^5$	3390.7
VDB 10%	1.18	$1.1*10^{10}$	$7.8*10^7$
VDB 50%	1.02	$1.1*10^{10}$	$7.7*10^7$
WJ	1.1	$9.7*10^8$	$3.2*10^7$

Conclusion and Future Work

- Approximate query processing over tuple bubbles
- Group data of relations into bubbles
- Compact but representative summaries
- Estimate results using summaries
- Future Work
  - Alternatives for bubbles creation
  - Combining results from single bubbles
  - Other approaches for summarization and combinations of approaches
  - Standard operators over tuple bubbles

Thank you for your attention!