Guest lecture. PluRel: Synthetic Data unlocks Scaling Laws for Relational Foundation Models (ICML 2026)

Stanford CS224W: Synthetic Relational Data from Structural Causal Models

CS224W: Machine Learning with Graphs

Vignesh Kothapalli, Stanford University

http://cs224w.stanford.edu

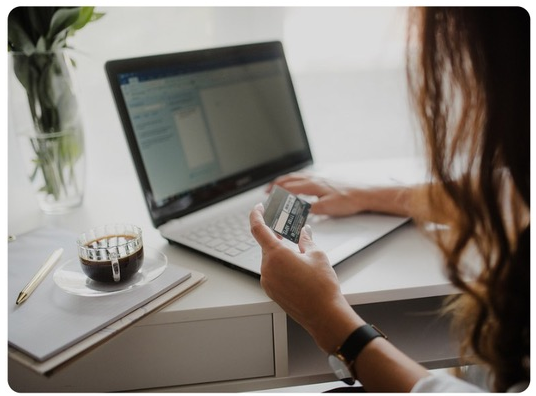


Lecture Outline

Motivation 10 min	**Tabular and relational foundation models, and the data bottleneck**	Tabular foundation models (TFMs) and relational foundation models (RFMs) in practice, and how they predict in context. Recap of relational deep learning. Why pretraining data is the bottleneck.
Part 1 25 min	**Structural causal models for a single table**	A running example from e-commerce. Structural equations and ancestral sampling. From one table to random SCMs: edge functions and a prior over SCMs.
Part 2 25 min	**Relational databases**	Schema graph. Foreign-key generation with a hierarchical stochastic block model. Structural equations that cross keys. Temporal consistency.
Part 3 15 min	**Pretraining and scaling laws**	Masked-cell pretraining of a Relational Transformer. Scaling in the number of databases and in tokens. Transfer to RelBench. Open problems.

Why Predict from Tables and Databases?

- Most operational data in commerce, finance and healthcare is stored in **relational databases**: tables linked by primary and foreign keys. The decisions that matter are **predictions** over these tables.



Fraud detection
Is this transaction fraudulent?



Recommendation
Which product will this customer purchase next?



Demand forecasting
What will this store sell next week?



Clinical risk
Will this patient be readmitted within 30 days?

- **Status quo:** each task is a separate engineering effort: **12.3 hours** of manual feature engineering per task, or **about 30 minutes** with relational deep learning.
- **Goal:** a single pretrained model that predicts on any database and any task, without task-specific training.

Photos (Wikimedia Commons): Shixart1985, CC BY 2.0; rawpixel, CC0; Ser Amantio di Nicolao, CC BY-SA 4.0; U.S. Army, public domain. Hours: RelBench user study, NeurIPS 2024.

Tabular Foundation Models Are Taking Off

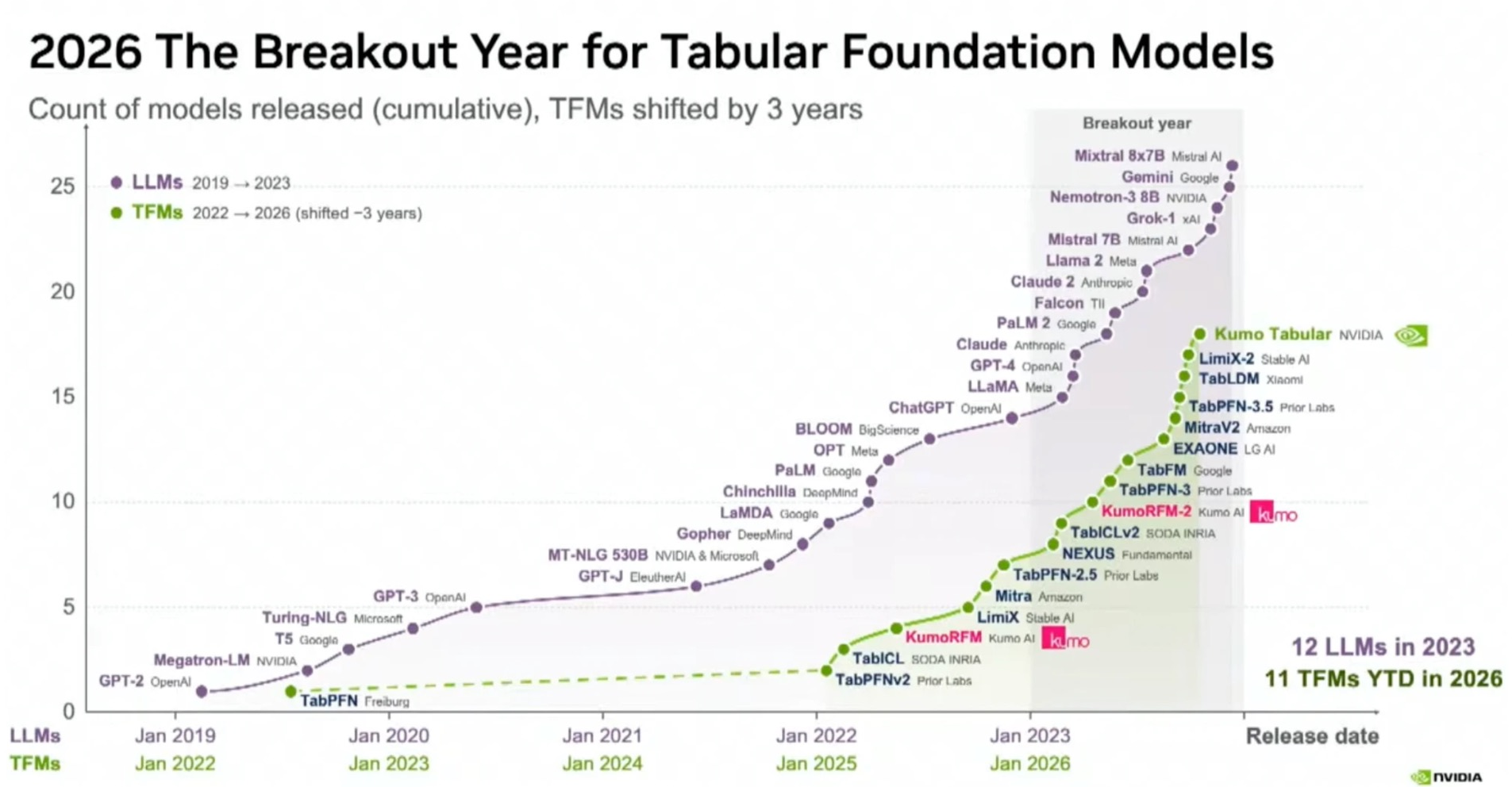


Figure: NVIDIA, 2026. Cumulative count of models released; tabular foundation model dates are shifted back by 3 years to line up with language models.

Tabular Foundation Models in Production, and in Orbit





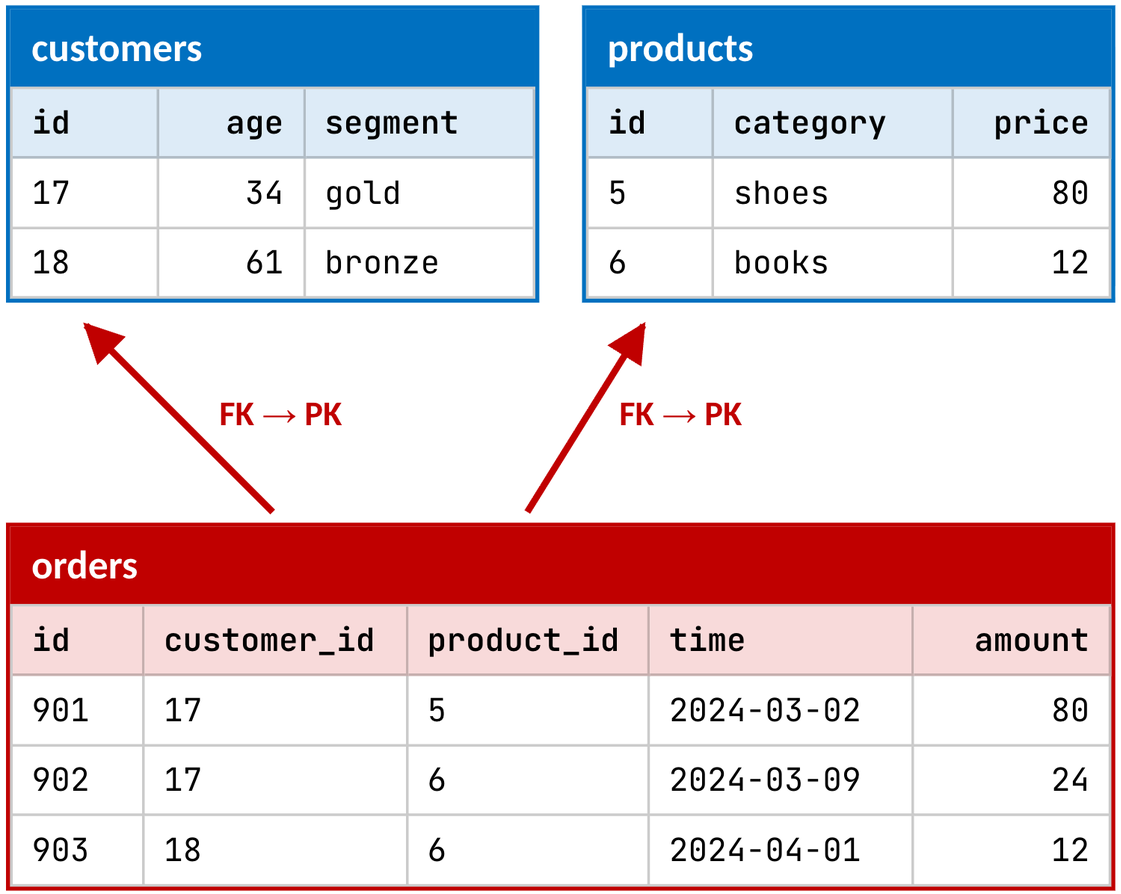
- **October 2026:** SAP will run its tabular foundation model, **SAP-RPT**, on board the **International Space Station** by early 2027.
  - It predicts failures from live telemetry and fills in missing sensor values, with no training or fine-tuning in orbit.
- **May 2026:** SAP agrees to acquire Prior Labs, the makers of TabPFN, and commits **more than €1 billion** over four years.

Yet a tabular foundation model reads a single table, while enterprise data spans many tables linked by keys.

Sources: SAP News Center, 27 October 2026 and 4 May 2026. Photo: NASA, STS-134, public domain.

Beyond a Single Table: Relational Databases

- A **relational database** is a set of tables. A **foreign key** holds the **primary key** of a row in another table.
- **Entity tables** describe what persists; **event tables** record what happens, with a timestamp.
- A **predictive task** asks for a label of an entity at time t, using only rows recorded before t.
- The foreign keys define a **graph over rows**: the relational entity graph.



Recap: Relational Deep Learning

- **Relational deep learning (RDL):** represent the database as a graph, with one node per row and one edge per foreign key, and train a GNN on it.

- **Strength:** no manual feature engineering; the joins are learned by message passing.

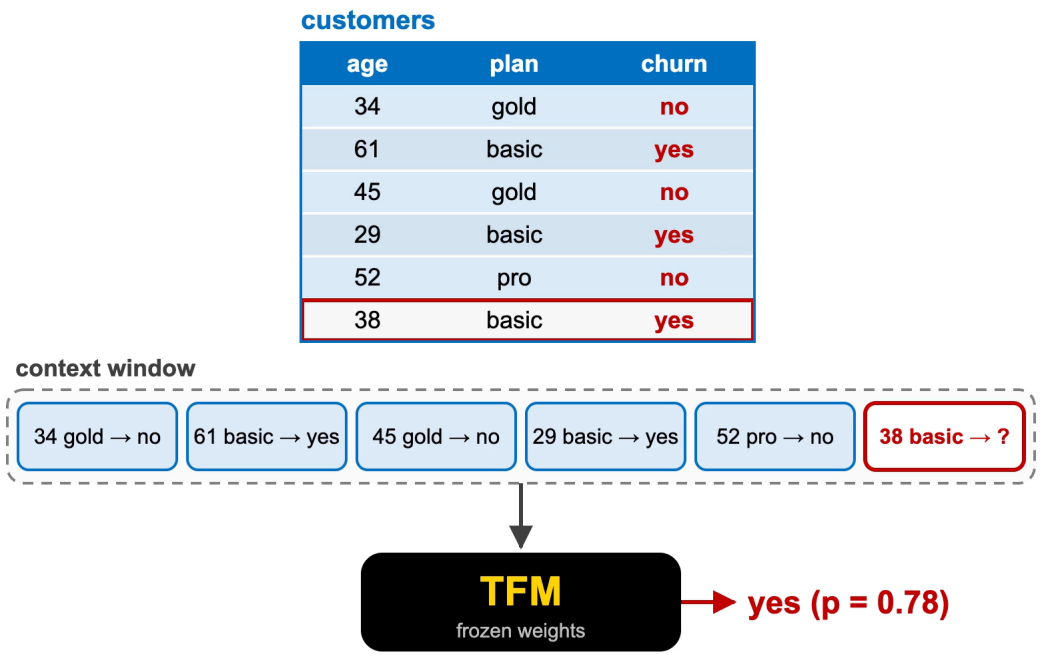
- **Limitation of the GNN:** one model per database and task, trained from scratch on that task's labels. Nothing transfers to the next database.

Can a single pretrained model make predictions on a database it has never seen?

In-Context Learning on a Table and on a Database

Tabular foundation model (TFM)

e.g. TabPFN: outperforms baselines tuned for 4 h, in 2.8 s

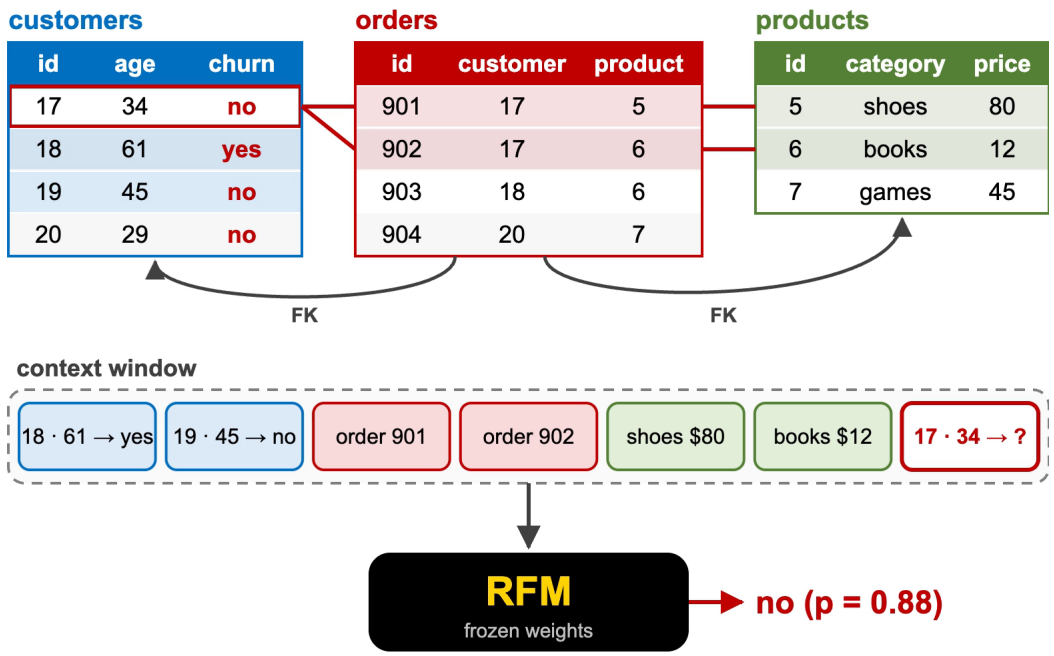


3 · Prediction: a single forward pass; the weights are not updated

- Context: labelled rows of the **same table**.

Relational foundation model (RFM)

e.g. Relational Transformer: 93% of supervised AUROC, zero-shot



3 · Prediction: a single forward pass; the weights are not updated

- Context: rows linked to the query by **foreign keys**.

The task is specified by the context, not by gradient updates. A TFM and an RFM differ only in how the context is assembled.

The Pretraining Data Bottleneck

- A relational foundation model must be pretrained on **many, diverse databases**: varied schemas, sizes, column types and dependencies.

Trillions

of tokens of public text for language models

Limited

public relational databases for relational models

- Real databases hold medical, financial and customer records: they are **private**, and no public corpus exists.

If real databases cannot be collected, we generate synthetic ones.

Part 1: Structural Causal Models for a Single Table

CS224W: Machine Learning with Graphs
Vignesh Kothapalli, Stanford University
http://cs224w.stanford.edu



Running Example: Customers of an E-Commerce Platform

A platform records one row per customer as the customer acts: sign-up, membership tier, visits and purchases. Whether the customer churns is observed 30 days later.

Platform event log

10:04	customer 1046 signs up, age 47
10:04	customer 1046 joins the bronze tier
10:04	customer 1046 visits the store (6 times this month)
10:04	customer 1046 buys running shoes, $30
+30 d	customer 1046 returns and buys again
10:20	customer 1047 signs up, age 29
10:20	customer 1047 joins the bronze tier
10:20	customer 1047 visits the store (1 times this month)
10:20	customer 1047 makes no purchase
+30 d	customer 1047 has no further activity

writes →

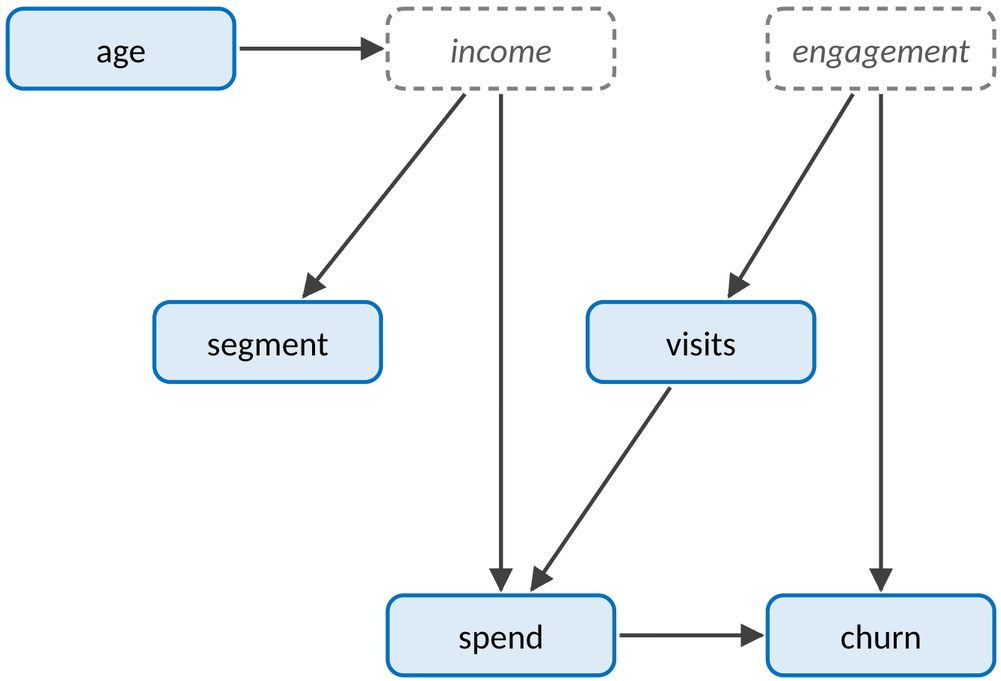
customer	age	segment	visits	spend	churn
1043	34	silver	12	$63	no
1044	61	gold	20	$155	no
1045	23	bronze	3	$12	yes
1046	47	bronze	6	$30	no
1047	29	bronze	1	$0	yes

One row per customer, written by the platform as events occur.

- This is the **ideal training data** for predicting churn: real behaviour, recorded as it occurs.

Modelling the Process That Generates the Data

- **Goal:** a method that generates tables resembling the platform's table: the same kinds of columns, and the same kinds of dependence between them.
- **Approach:** model the **process** that produces the data: which variables influence which, and how.
  - Some causes are never recorded: income influences both the segment and the spend.
- A **structural causal model (SCM)** makes this model executable: each variable is a function of its direct causes and independent noise.
- Sampling the SCM yields **synthetic rows** whose dependence structure matches that of the real table.



Solid: recorded column. Dashed: latent cause, never recorded.

Structural Causal Models

Definition. A structural causal model is a triple (G, F, P) where

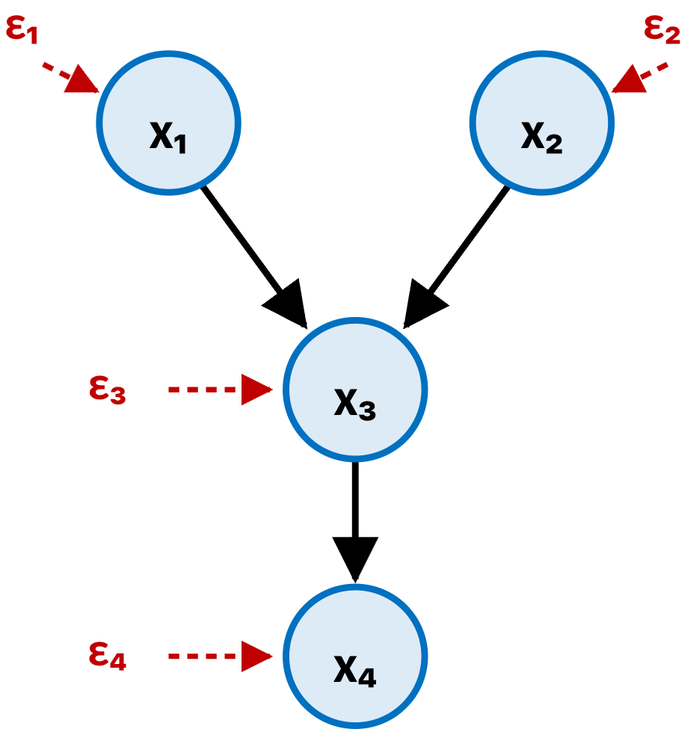
G is a directed acyclic graph over variables x_1, …, x_d

F is a set of structural equations, one per variable:

$$x_j = f_j(\text{parents}(x_j), \varepsilon_j)$$

P is a distribution of independent exogenous noise ε_1, …, ε_d

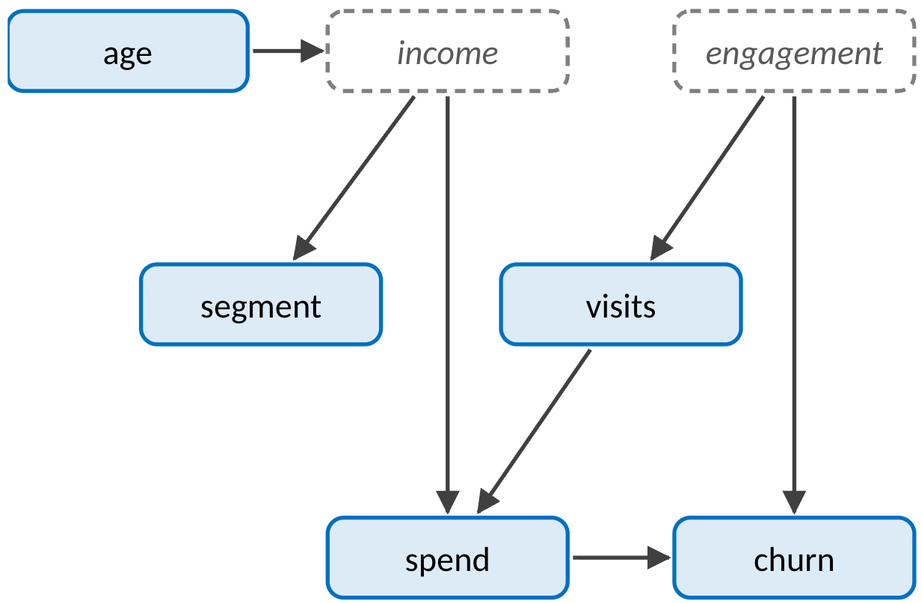
- **Ancestral sampling:** draw ε, then evaluate the variables in a topological order of G. One pass gives one row.
- **Table:** n independent passes give n rows; the d variables are the columns.
- **Joint distribution:** $p(x) = \prod_j p(x_j \mid \text{parents}(x_j))$, sparse by construction.



$x_3 = f_3(x_1, x_2, \varepsilon_3)$ $x_4 = f_4(x_3, \varepsilon_4)$

Worked Example: An SCM for the Customers Table

The five columns of the running example, with two latent causes. The equations act on latent values; each recorded column is then mapped onto its marginal distribution.



$\texttt{age} = \varepsilon_1$

Uniform(18, 80), whole years

$\texttt{income} = \texttt{age} + \varepsilon_2$

latent: never recorded

$\texttt{engagement} = \varepsilon_3$

latent: never recorded

$\texttt{segment} = \texttt{income} + \varepsilon_4$

cut into bronze 60%, silver 30%, gold 10%

$\texttt{visits} = \texttt{engagement} + \varepsilon_5$

Poisson(6): a count per month

$\texttt{spend} = \texttt{income} + \texttt{visits} + \varepsilon_6$

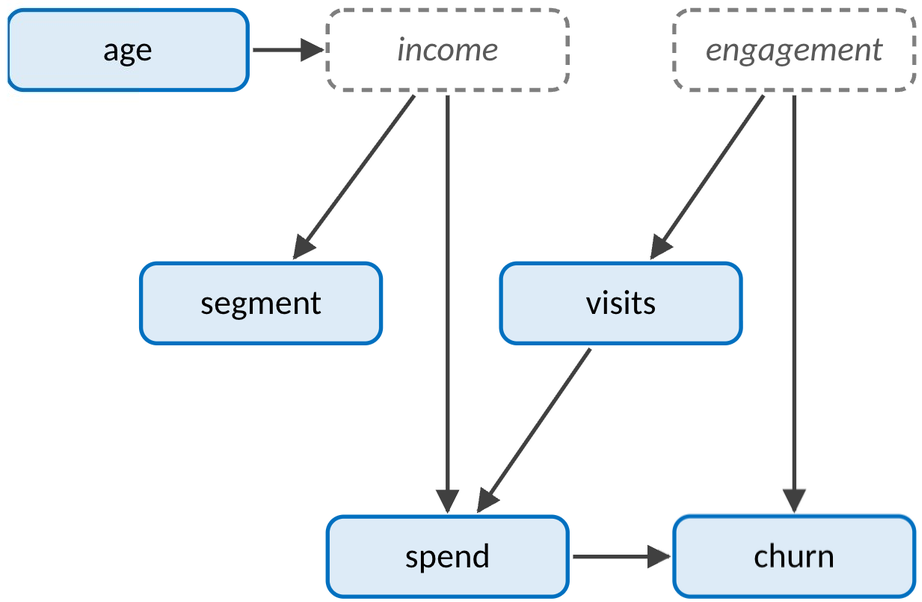
LogNormal(3, 0.5): positive, right-skewed

$\texttt{churn} = -\texttt{engagement} - \texttt{spend} + \varepsilon_7$

cut into no 70%, yes 30%

Ancestral Sampling

Synthetic rows for the running example. Each click evaluates one structural equation; its result then enters the table.



Income and engagement are evaluated and used by their children, but never enter the table.

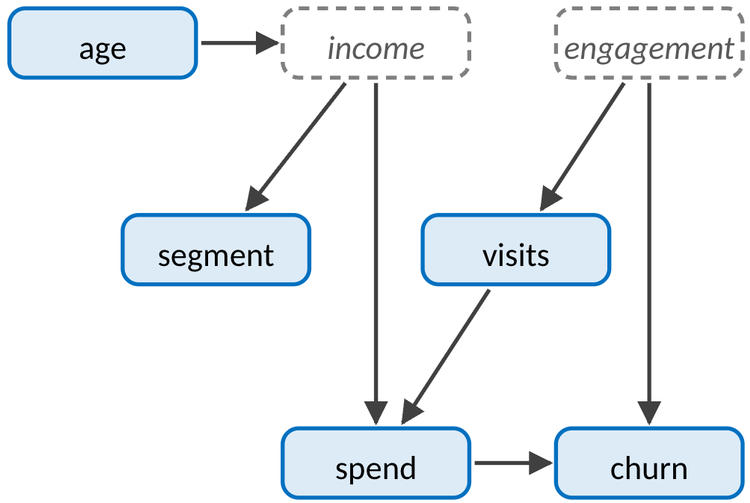
age	segment	visits	spend	churn
63	silver	7	$29	no
27	bronze	4	$12	yes
78	gold	9	$70	no
62	silver	9	$48	no
48	bronze	9	$31	no

Structural equations for row 1, evaluated in topological order

```
age = ε₁                          = +0.80 → 63
income = age + ε₂                 = +0.70 (latent)
engagement = ε₃                   = +0.60 (latent)
segment = income + ε₄             = +1.20 → silver
visits = engagement + ε₅          = +0.80 → 7
spend = income + visits + ε₆      = +1.60 → $29
churn = -engagement - spend + ε₇  = -2.40 → no
```

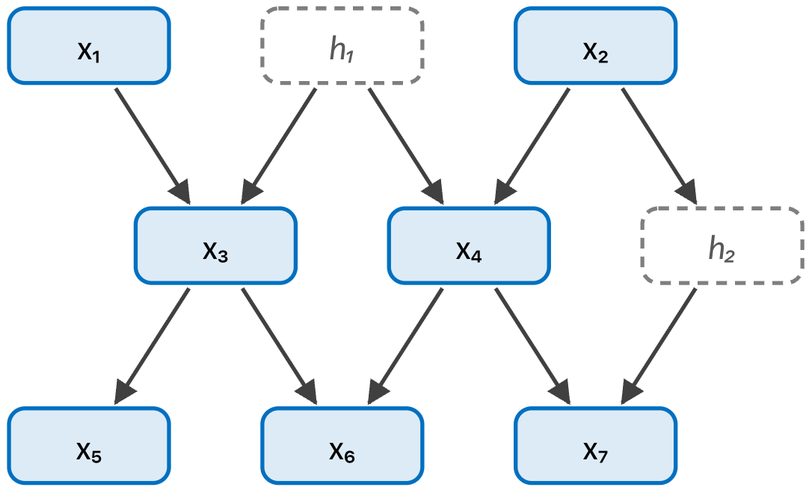
From the Running Example to Any Table

The running example



drop the names,
draw at random

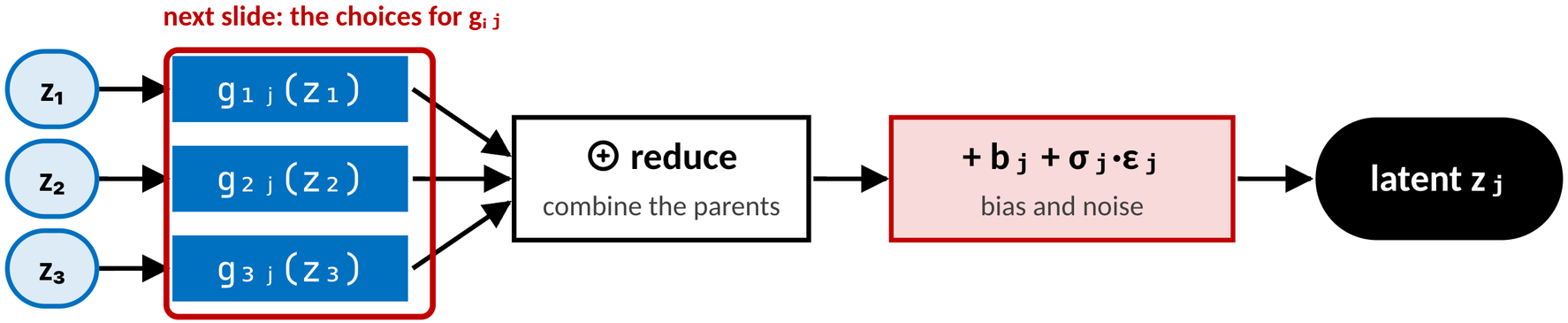
A random SCM



- Keep the form of the model, drop the names: **d generic variables** with no fixed meaning.
- Draw every ingredient at random: the **graph**, the **function** on each edge, the **noise** levels, and which variables are **recorded, categorical or latent**.
- Each draw is a different table produced by a different mechanism. The next slides make each ingredient precise.

The Structural Equation of a Node

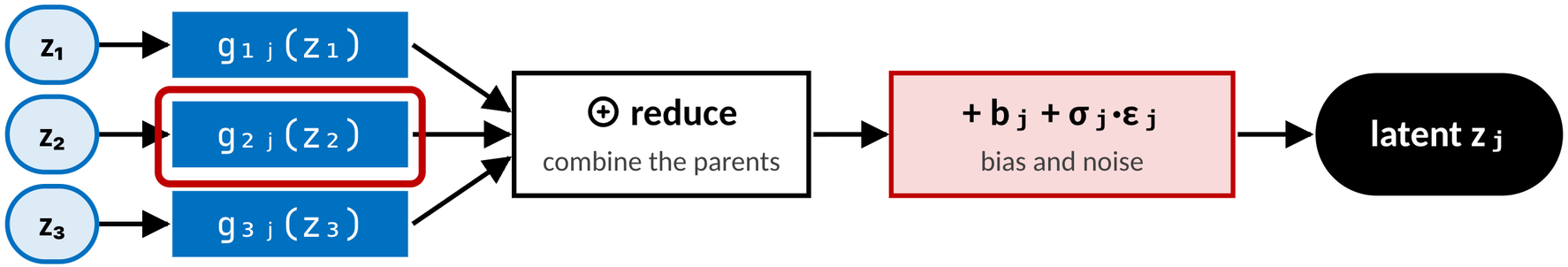
$$z_j = \oplus_i g_{ij}(z_i) + b_j + \sigma_j \cdot \varepsilon_j$$



Step	Purpose	Possible choices
Edge functions	A nonlinear effect of each parent	linear, MLP, tree, Fourier, quadratic, …
Reduction ⊕	How the parents combine	sum, product, max, min, …
Bias and noise	What the parents do not explain	any noise level σ_j; ε normal or heavy-tailed

- **Running example:** every $g_{ij}(z) = z$, ⊕ is a sum and $b_j = 0$, e.g. spend = income + visits + ε_6.

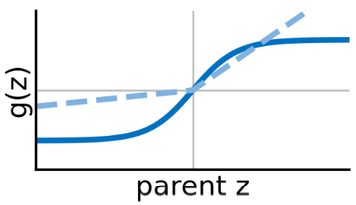
Edge Function Families



The highlighted edge function g_{2j} can take any of these shapes:

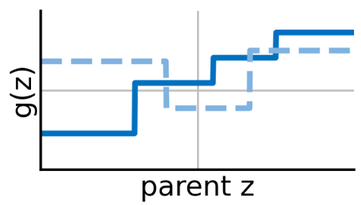
Each panel: g(z) against the parent z. **Solid** and dashed: two random draws.

Linear $w \cdot \varphi(z)$



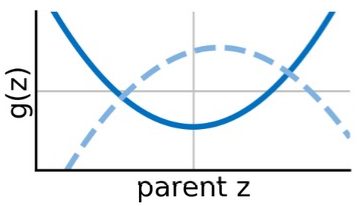
Trends and saturation; includes g(z) = z

Tree average of trees



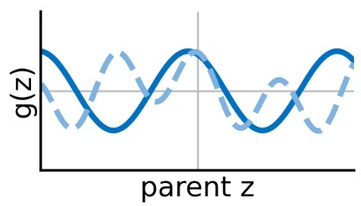
Threshold rules: steps

Quadratic $a \cdot z^2 + b \cdot z + c$



Peaks and valleys

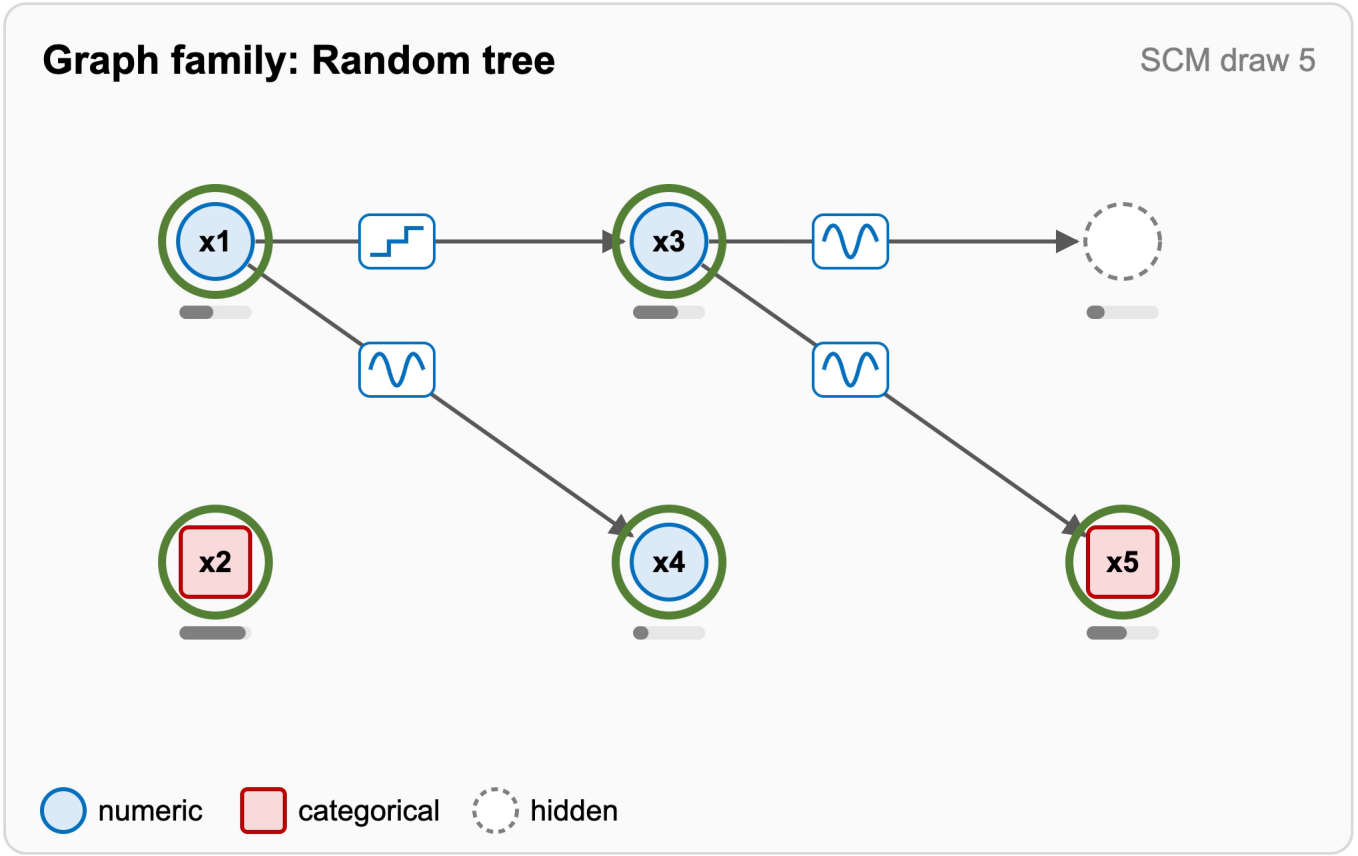
Fourier $\Sigma \cos(\omega z + b)$



Periodic effects

A Prior over Structural Causal Models

From one node to a whole SCM. A prior draws the entire SCM at random, one ingredient at a time.



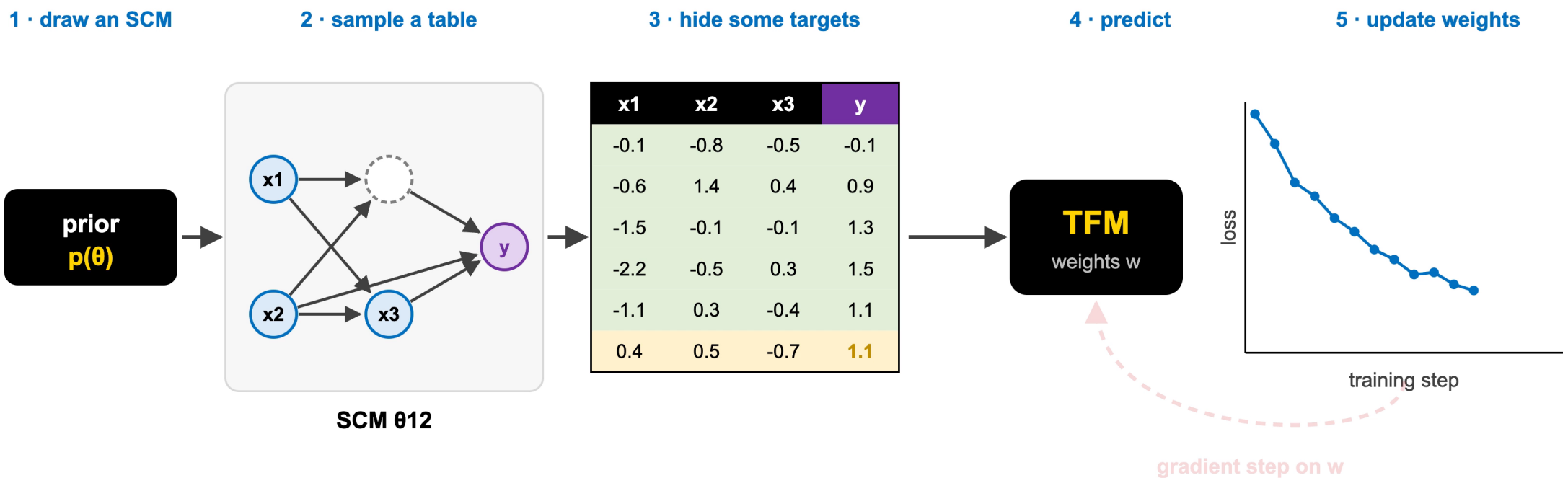
1. **Graph**
   a random-graph family, size and density
2. **Node types**
   numeric, categorical or hidden
3. **Edge functions**
   a family for every edge
4. **Noise**
   a noise level σ per node (grey bar)
5. **Table**
   recorded nodes become columns

x1	x2	x3	x4	x5
-1.3	C	0.5	0.0	C
-0.1	C	-0.1	1.1	A
-1.3	B	1.0	0.2	C

Each draw is a new SCM, and hence a new table.

Pretraining a Tabular Foundation Model on Synthetic Tables

Every training step draws a fresh SCM from the prior, samples a table, hides the target of some rows and trains the model to predict it from the rest.



training step 12: a new SCM, a new table

- The model never sees real data during training: it learns to predict **in context** across millions of synthetic tables.
- At test time a real table is the context, with no gradient steps. This is how TabPFN is trained; Part 2 does the same for **databases**.

Part 2: From a Single Table to a Relational Database

CS224W: Machine Learning with Graphs
Vignesh Kothapalli, Stanford University
http://cs224w.stanford.edu



Running Example: The Same Platform as a Database

The same events, recorded as a **relational database**: one table per kind of entity or event, linked by foreign keys.

Platform event log

09:02	customer 1043 signs up, age 34
09:05	1043 buys headphones, $48
09:15	customer 1044 signs up, age 61
09:21	1044 buys an espresso machine, $120
09:24	1044 buys coffee beans, $35
09:31	customer 1045 signs up, age 23
09:40	1045 buys headphones, $48

customers

id	age	segment
1043	34	silver
1044	61	gold
1045	23	bronze

products

id	category	price
5	electronics	$48
6	kitchen	$120
7	grocery	$35

FK FK

orders

id	customer_id	product_id	time	amount
901	1043	5	09:05	$48
902	1044	6	09:21	$120
903	1044	7	09:24	$35
904	1045	5	09:40	$48

Each event is a row in the table it describes; foreign keys link the rows.

- Richer than one table: every order is kept, with its time and its product. Churn becomes a prediction over time: will customer 1045 order again after time t?

Why Not Flatten to a Single Table?

Two ways to flatten; both lose information.

customers

id	age	segment
1044	61	gold

orders

id	customer_id	product_id	time	amount
902	1044	6	09:21	$120
903	1044	7	09:24	$35

Option A: join orders with customers

order	product	time	amount	age	segment
902	6	09:21	$120	61	gold
903	7	09:24	$35	61	gold

- Rows are orders: **customers without orders vanish**.

Option B: aggregate orders per customer

id	age	segment	n_orders	total
1044	61	gold	2	$155

- Rows are customers: **individual orders and their timing are lost**.

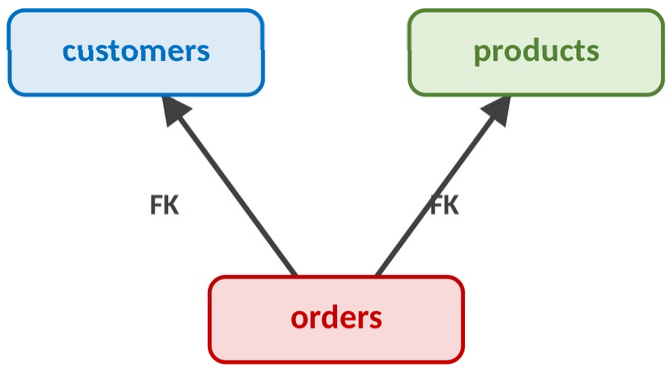
Key idea: the links are data; they must be generated, not flattened away.

Modelling a Database: Three Graphs

Generating a database means sampling **three graphs**, one per level, in this order. Running example:

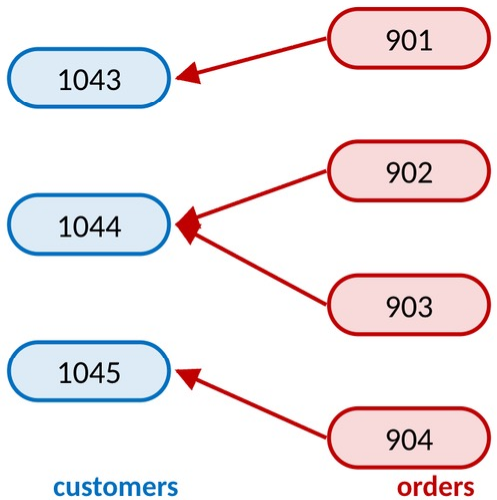
Stage 1 · Schema graph

which tables exist, and which reference which



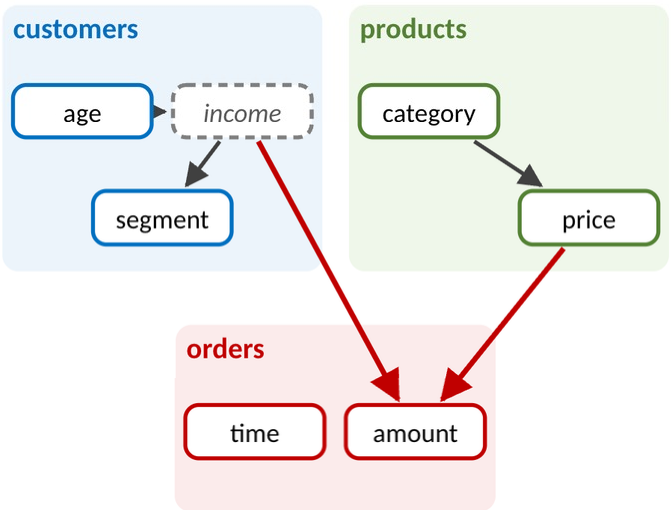
Stage 2 · Key graph

which row points to which row



Stage 3 · Column graph

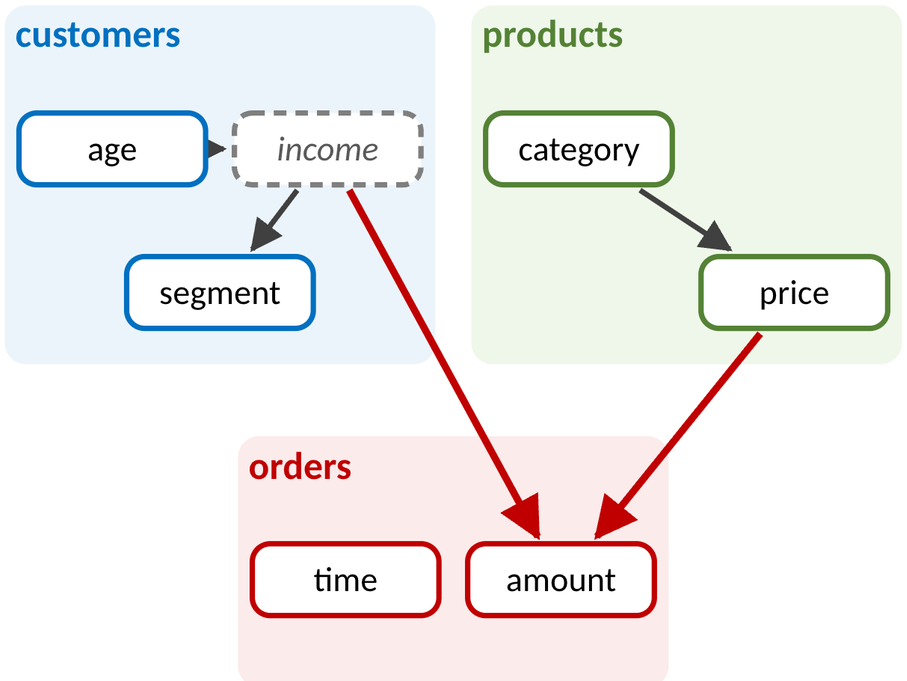
how values depend, within and across tables



- Stage 1 fixes the tables, Stage 2 the links between rows, Stage 3 the values in every cell.
- A Stage 3 edge can cross a foreign key (red): an order reads the income of its customer and the price of its product.

Worked Example: An SCM for the Database

The SCM of Part 1, extended to three tables. Each table has its own equations; the red edges read a column of the row that a foreign key points to.



customers

```
age = ε₁
income = age + ε₂ (latent)
segment = income + ε₃
```

products

```
category = ε₄ (cut into categories)
price = category + ε₅
```

orders

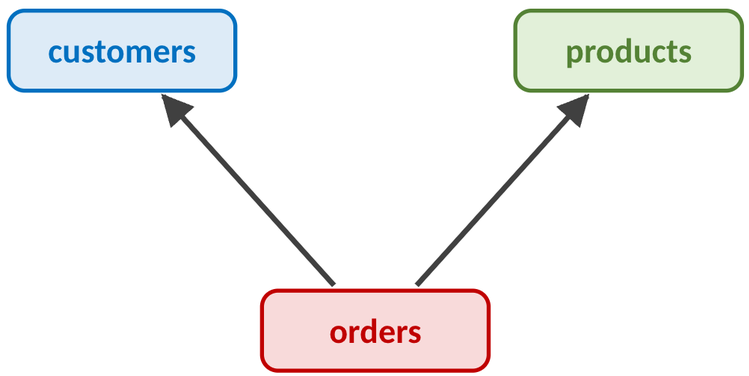
```
customer_id, product_id (sampled in Stage 2)
time = ε₆ (a calendar)
amount = income[customer_id] + price[product_id] + ε₇
```

- income[customer_id]: the income of the customer this order points to. Two orders of the same customer share it, so rows are no longer independent.

Stage 1: The Schema Graph

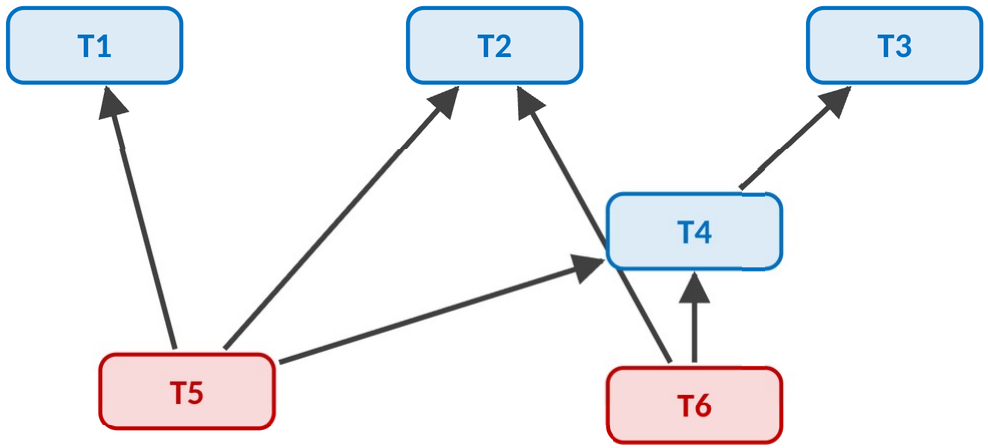
Stage 1 samples a directed acyclic graph over tables: an edge is a foreign key, from the referencing table to the referenced one.

Running example



draw at random

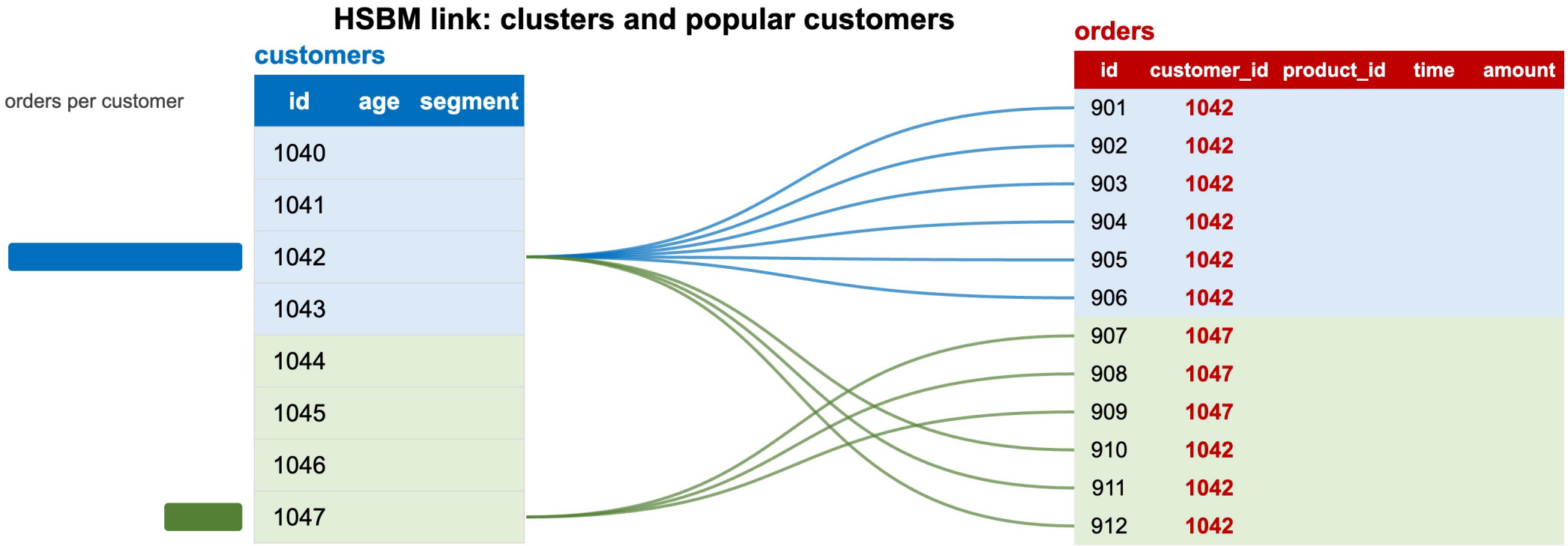
A random schema



- **Static tables** are referenced by some other table: entities such as customers and products, with no timestamp.
- **Event tables** are referenced by none: activities such as orders, each row with a timestamp.
- The role is not chosen; it **follows from the graph**. The graph itself is drawn from the random-graph families of Part 1.

Stage 2: Linking the Rows

Each order picks the customer its key points to.

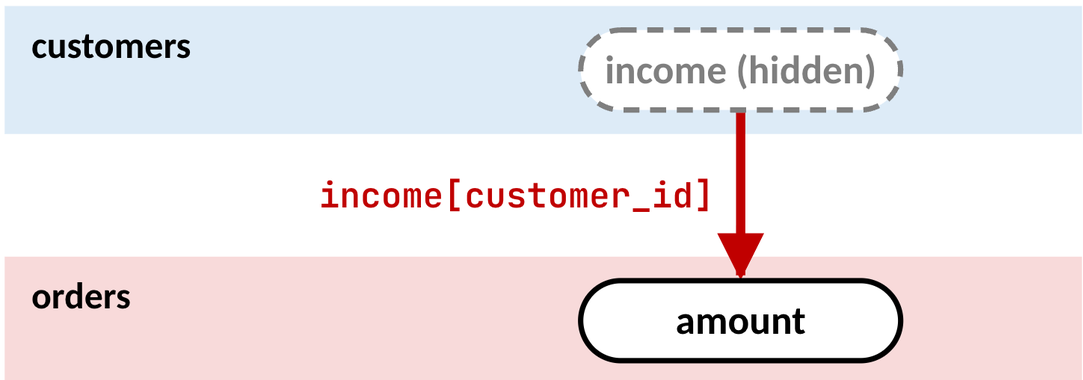


- Real keys have **communities** and a few **very popular** customers; the HSBM link reproduces both.

Stage 3: Reading across Foreign Keys

Some edges cross a foreign key: an order reads columns of the rows it points to.

An order reads its customer



Example. Customer 1044 has income latent 1.3. Its orders 902 and 903 both read 1.3: higher-income customers place larger orders.

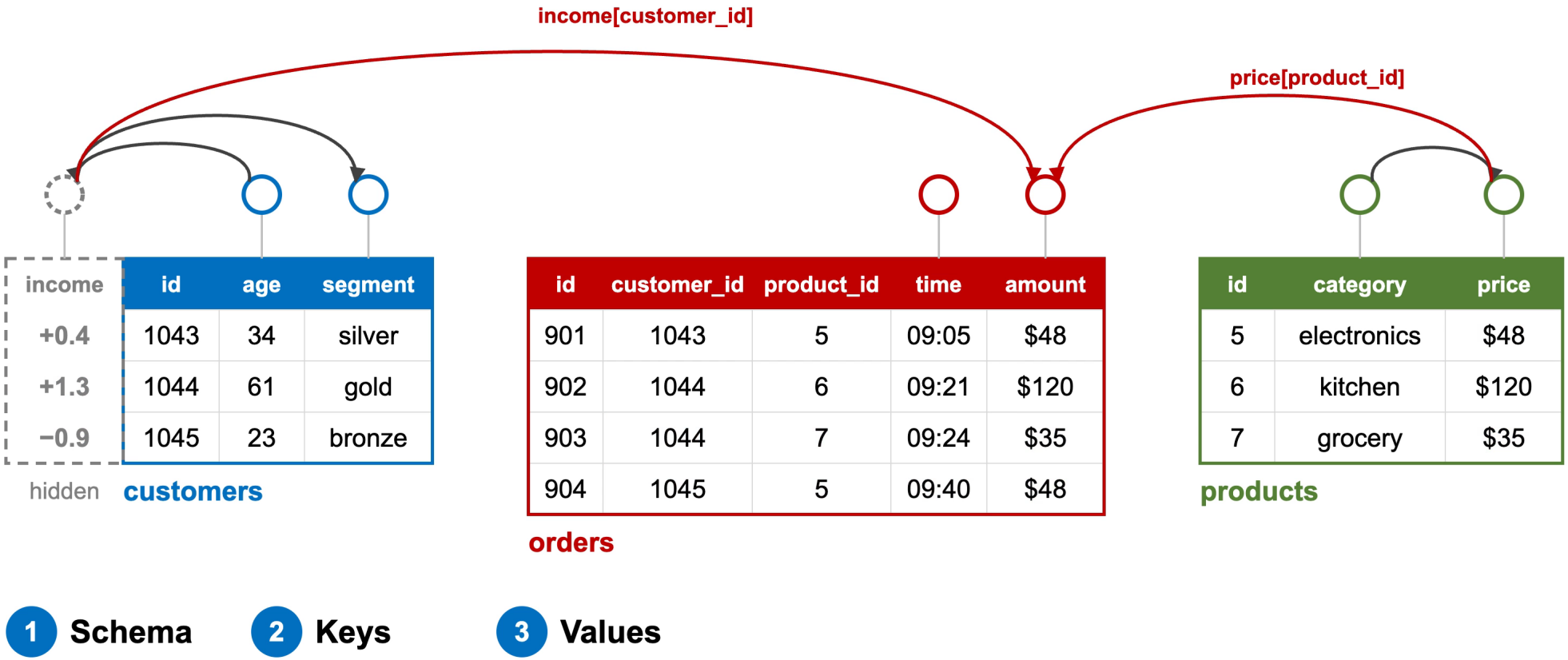
An order reads its product



Example. Order 902 points to product 6 and reads its price, $120.

Both are one hop of message passing along a key, as in a GNN layer.

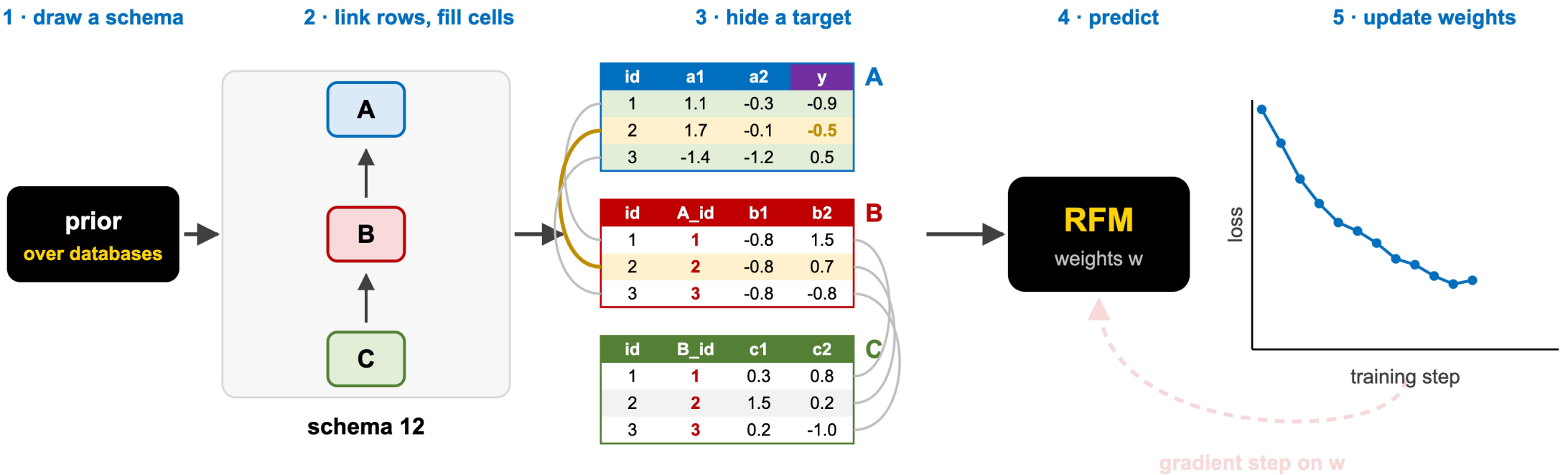
The Three Stages on the Running Example



- Static tables are filled first; each order then reads the rows its keys point to.

Pretraining an RFM on Synthetic Databases

The recipe of Part 1, one level up: every training step draws a new schema, links its rows, fills its cells, hides a target and trains the model to predict it.



training step 12: a new schema, a new database

- The context of a prediction is the **linked rows**: the query row and the child rows that point to it.
- No real database is used in training; at test time a real database is the context. This is how PluRel pretrains the Relational Transformer.

Part 3: Pretraining and Scaling Laws

CS224W: Machine Learning with Graphs
Vignesh Kothapalli, Stanford University
http://cs224w.stanford.edu



The Relational Transformer: One RFM

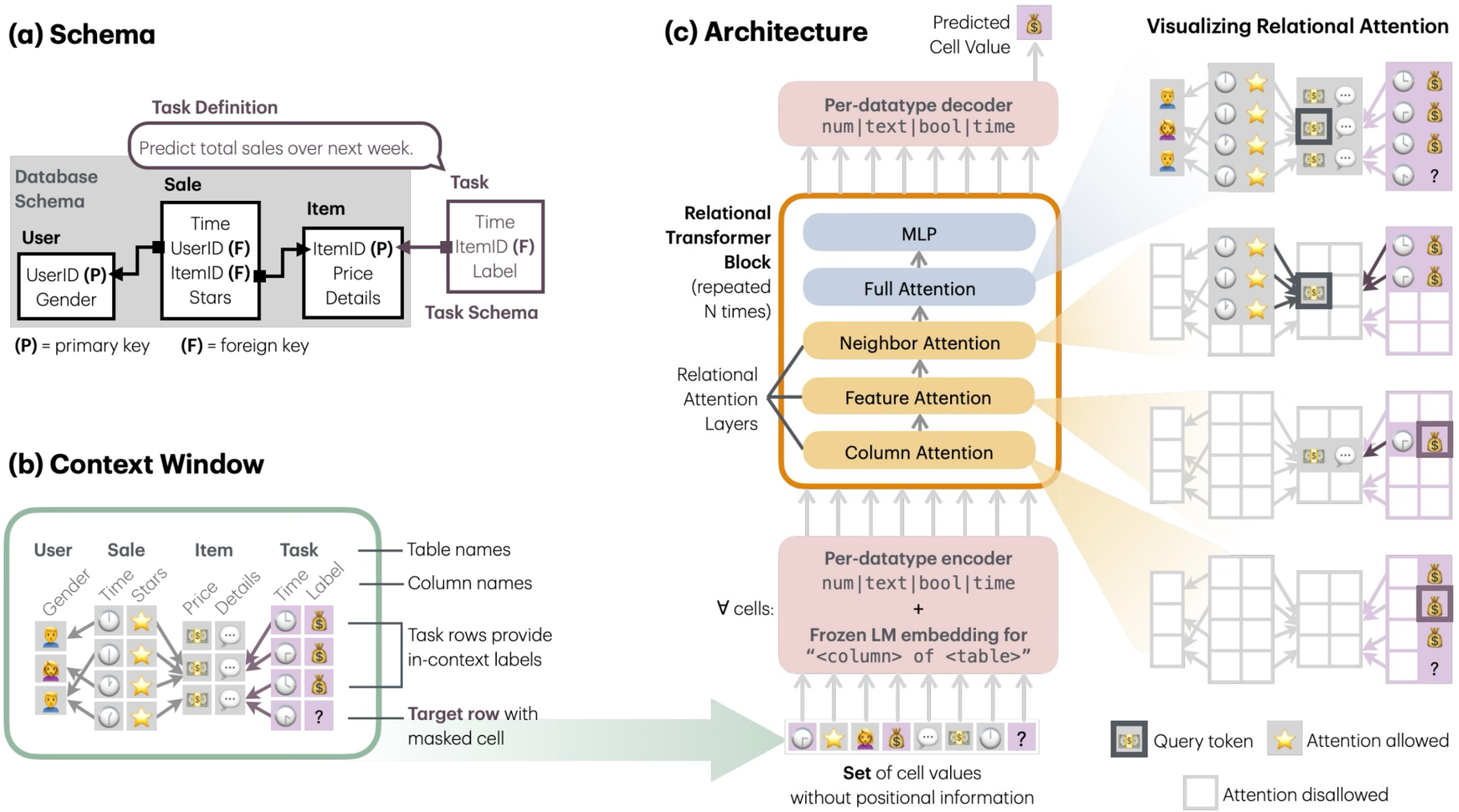
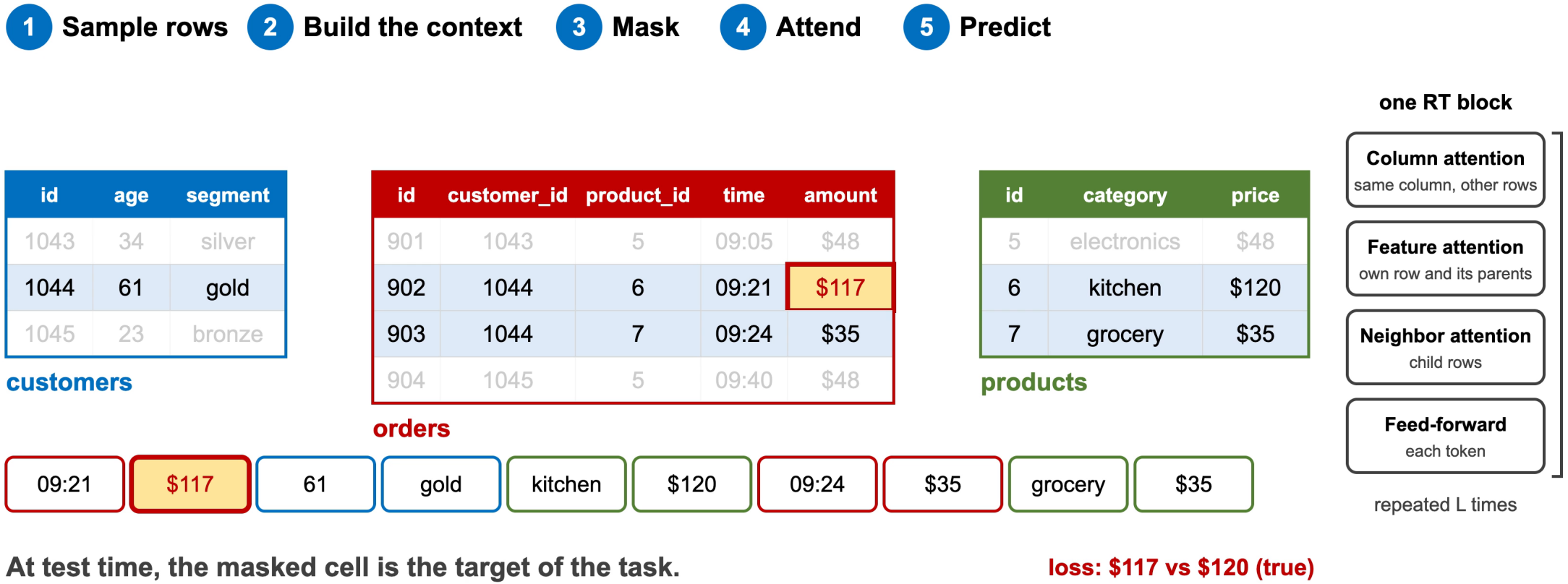


Figure from Ranjan et al., Relational Transformer: Toward Zero-Shot Foundation Models for Relational Data, ICLR 2026.

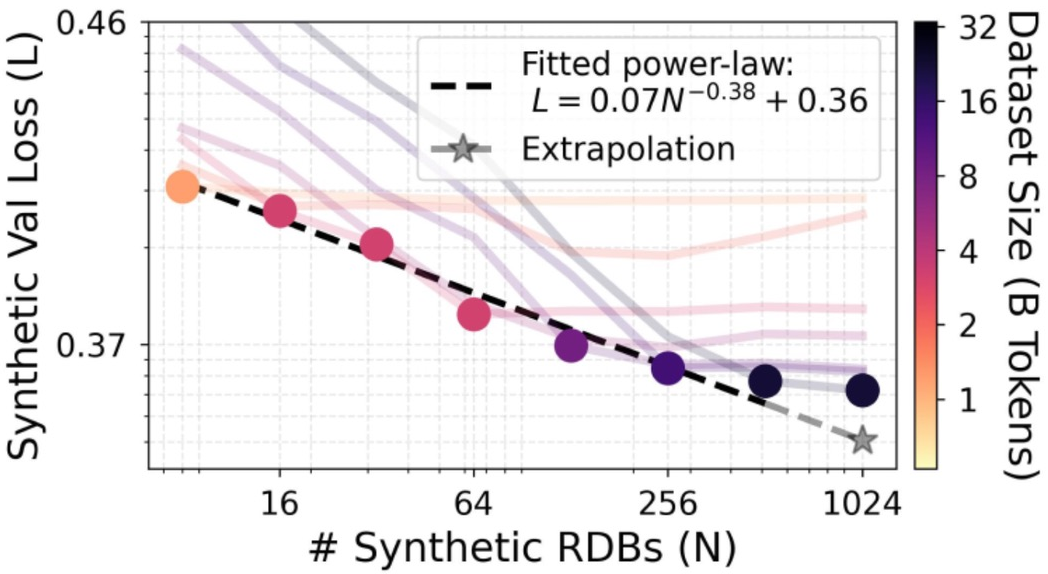
Relational Attention and Masked Cell Prediction

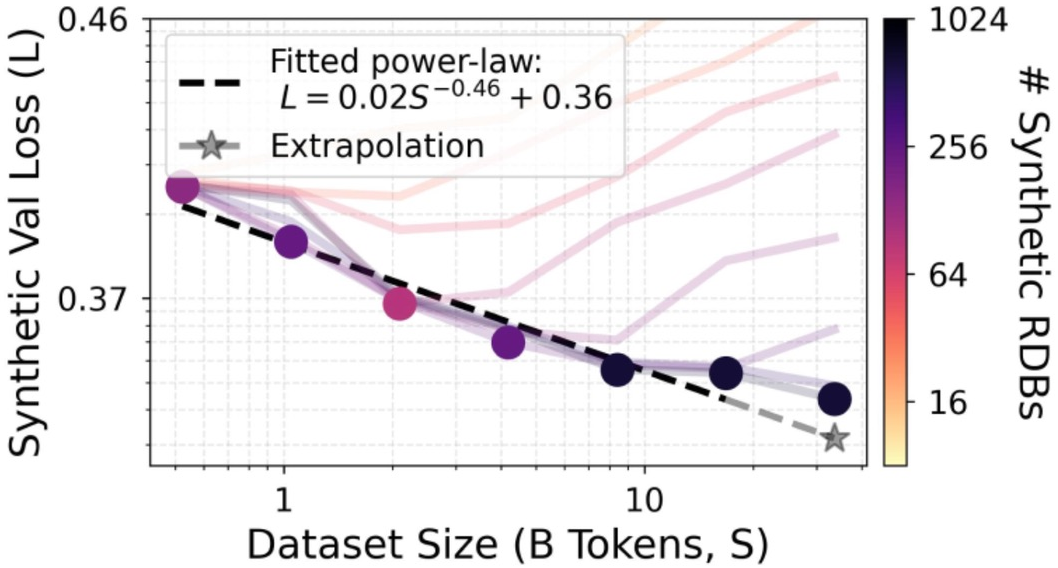


- On PluRel, every step draws a new database, seed row and mask; next: how this loss scales.

A single masked cell, exactly as in a prediction task. Predicted value illustrative.

Result 1: Power-Law Scaling of Pretraining Loss





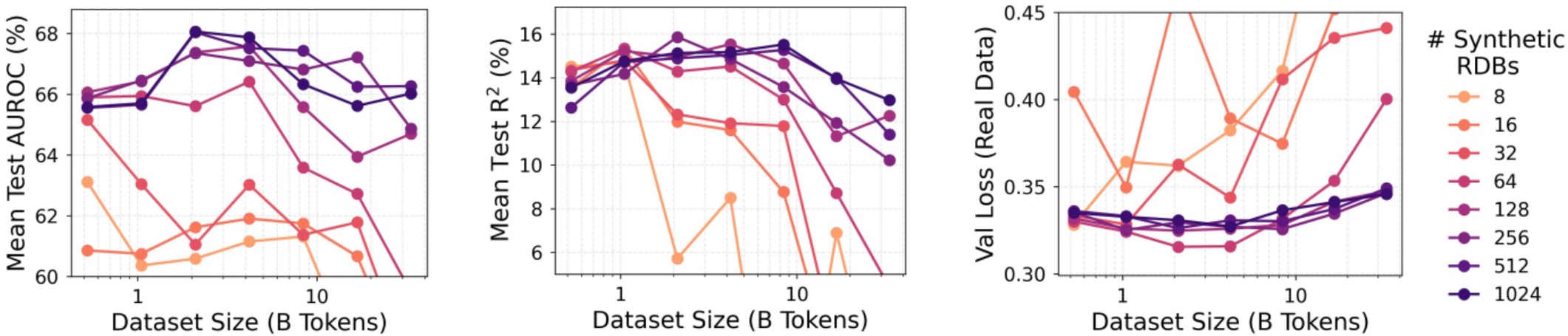
N = number of synthetic databases, 8 to 1,024

S = pretraining tokens, 0.5B to 32B

Loss falls as a power law in both: more databases and more tokens keep helping.

Kothapalli et al., PluRel, ICML 2026. Loss on held-out synthetic databases; faint lines are individual runs.

Result 2: Zero-Shot Transfer Requires Diversity



(a) Mean 0-shot test AUROC (%) (↑)

(b) Mean 0-shot test R^2 (%) (↑)

(c) Validation loss (↓) on real data (RelBench)

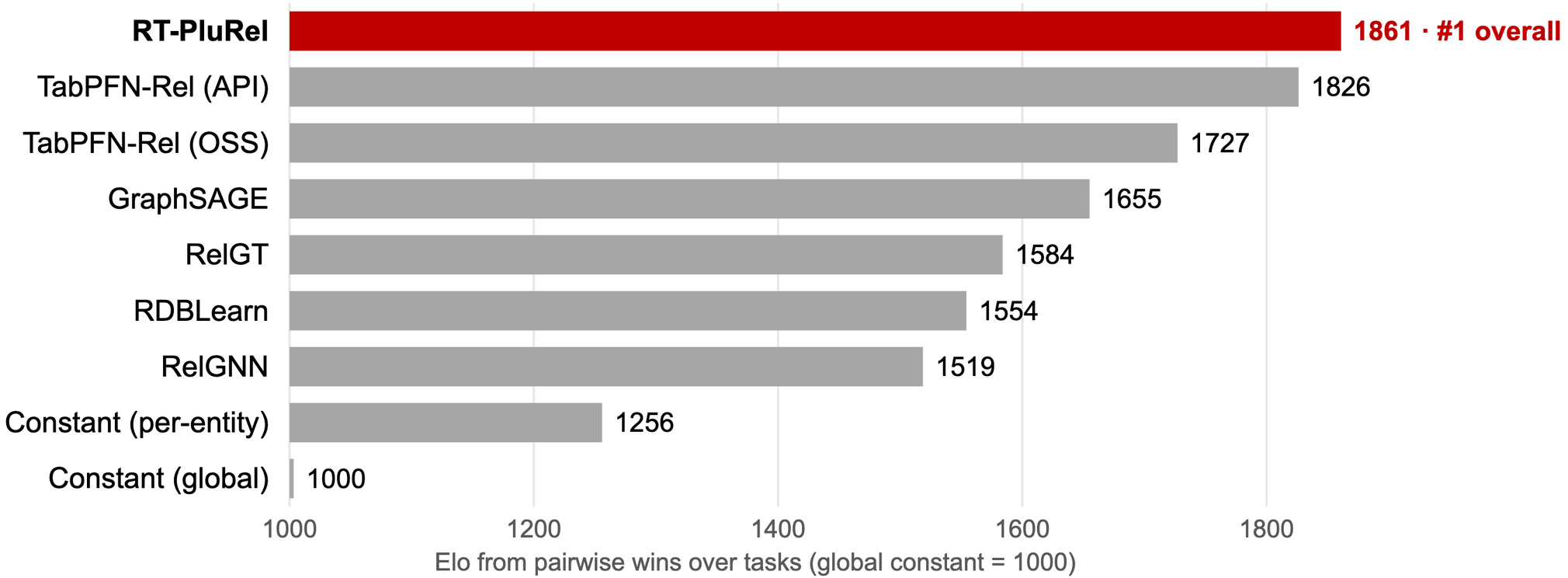
8–32 databases: more tokens make real-data results *worse*. The model overfits to a few data-generating mechanisms.

Hundreds of databases: more tokens help. Zero-shot transfer to RelBench works.

Zero-shot: pretrained only on synthetic data, evaluated on real RelBench tasks.

Result 3: RT-PluRel Is #1 on RelArena-α

RelArena-α re-runs every method under one protocol: 21 forecasting tasks on 7 databases.



Pretrained only on synthetic databases, then fine-tuned per task: first end to end.

RelArena-α (Prior Labs and STAR), first leaderboard, combined model + system board, single seed.

Summary

1 **The pretraining prior determines the inductive bias.**

Without large public relational corpora, the data generator defines what a relational foundation model learns.

2 **Structural causal models yield realistic dependence.**

Each cell is a function of its causes, including unobserved latents, plus independent noise.

3 **A relational database is generated from three graphs.**

Schema, keys and columns; dependence across foreign keys corresponds to message passing.

4 **Pretraining loss scales with the number of databases.**

The loss follows a power law in distinct databases; with too few, additional tokens degrade transfer.

5 **Synthetic pretraining is an effective initialization.**

RT-PluRel, pretrained solely on PluRel and fine-tuned per task, ranks first on RelArena-α.

Thank You! Questions?

Paper · arxiv.org/abs/2602.04029

Code · github.com/stanford-star/plurel

Project · star-project.stanford.edu/plurel

Data and models · huggingface.co/stanford-star

With Rishabh Ranjan, Valter Hudovernik, Vijay Prakash Dwivedi, Johannes Hoffart, Carlos Guestrin and Jure Leskovec



Backup Slides

CS224W: Machine Learning with Graphs
Vignesh Kothapalli, Stanford University
http://cs224w.stanford.edu



PluRel API: The Running-Example Database

The running-example database in PluRel 2. A **ForeignKey** names its parent table and how rows are linked; **"customer_id.income"** reads the income of the customer the key points to.

```
customers = Table({"id": PrimaryKey(), "age": Numeric(Uniform(18, 80), round=1),
            "income": Hidden(causes={"age": 1.0}),
            "segment": Categorical({"bronze": 0.6, "silver": 0.3, "gold": 0.1}, causes={"income": 1.0})})
products = Table({"id": PrimaryKey(), "category": Categorical({"grocery": 0.5, "kitchen": 0.3, "electronics":
0.2}),
            "price": Numeric(LogNormal(3, 0.7), causes={"category": {"kitchen": 1.0, "electronics": 1.5}})})
orders = Table({
    "id": PrimaryKey(),
    "customer_id": ForeignKey("customers", link=HSBMLink(popularity=Pareto(2.5))),
    "product_id": ForeignKey("products"),
    "time": Timestamp("2024-01-01", "2025-01-01"),
    "amount": Numeric(LogNormal(3, 0.5), causes={"customer_id.income": 1.0, "product_id.price": 1.0}),
}, time_column="time")
db = Schema({"customers": customers, "products": products, "orders": orders})
frames = db.sample({"customers": 500, "products": 100, "orders": 5000}, seed=0)
```

Measured on this draw: Spearman(amount, price) = 0.50; median price grocery $12, kitchen $30, electronics $48; median order bronze $15, gold $30.

Masked-Cell Pretraining of a Relational Transformer

To evaluate the prior we need a model that reads any database: the Relational Transformer (RT). Its objective is the PFN loss, with databases in place of tables:

$$L(\theta) = \mathbb{E}_{D \sim p(D),\ \text{mask}} \left[-\log q_\theta(\text{masked cell} \mid \text{visible cells of } D) \right]$$

orders, with one cell masked

time	customer_id	amount
Mar 2	17	40
Mar 9	17	?
Apr 1	18	95

Each cell is one token, built from its value, its column name and its table name. No schema-specific parameters.

- **Relational attention.** A cell attends to cells of its row, cells of its column, and cells of rows linked by a foreign key.
- **One objective for every task.** A churn label or a sales value is one more cell to predict; classification and regression share the model.
- **Protocol.** Pretrain on N synthetic databases with a budget of S tokens, then evaluate zero-shot on RelBench databases the model has never seen.

Relational Transformer: github.com/stanford-star/relational-transformer

Hierarchical Stochastic Block Model for Keys

Each child row c draws its parent row p independently, with probability

$$P(\ c \to p\ ) \propto w_p \cdot \prod_\ell A_\ell[\ b_\ell(p),\ b_\ell(c)\ ]$$

- **$b_\ell(\cdot)$**: cluster label of a row at level ℓ of a hierarchy of clusters.
- **A_ℓ**: block affinity: high for matching clusters, near zero otherwise. This gives **communities**.
- **w_p**: popularity of a parent row, drawn from a heavy-tailed distribution. This gives **hubs**.
- **Inactive parents**: a share of parent rows has $w_p = 0$ and receives no children.

Measured: 500 customers, 5,000 orders (seed 0)

Orders per customer	Uniform	Pareto popularity
Mean	10	10
Median	10	4
Maximum	20	1,041
Customers with no orders	0	67
Orders held by the top 10%	16%	56%

- **Why it matters for a GNN or a transformer:** neighbourhood sizes vary over three orders of magnitude, as in real data. A uniform link never shows the model a hub or an isolated entity.

One Database Drawn from the Prior

Actual output of SchemaPrior().realize(seed=3): 4 tables, 4 foreign keys, 46,479 rows.

Schema

Table	Rows	Role	Keys to
t0	536	static	none
t1	775	static	none
t2	17,879	event	t0, t1, t1
t3	27,289	event	t0

First rows of event table t2

id	col0	col1	col2	time	t0_id	t1_id	t1_id2
0	c5	-1.997	0.016	1990-01-02 20:23	493	671	572
1	c0	1.373	1.507	1990-01-03 13:44	345	294	338
2	c5	-0.278	0.959	1990-01-05 12:10	76	430	311
3	c1	-0.638	1.615	1990-01-06 01:24	14	613	269

Structural equation of one hidden node of t2, as printed by explain()

```
n2: hidden
  ← n0 (QuadraticEdge) # a node of the same table
  ← 0.841 * t0_id.n2 (square) # node n2 of the t0 row that t0_id points to
  ← t1_id.n13 (FourierEdge) # a node of the t1 row
```

The Prior over Databases

Every structural choice is a random variable with a range. One seed fixes all of them and yields one database.

Stage	Random variable	Default range
Schema	Number of tables	2 to 20, log-uniform (median 7 over 40 draws)
Schema	Schema graph family	Barabási–Albert, random tree, reverse tree, Watts–Strogatz, layered, Erdős–Rényi
Schema	Rows per table	500 to 1,000 for static tables; 10,000 to 30,000 for event tables
Keys	HSBM parameters	1 to 5 levels, 1 to 3 clusters each, Pareto popularity, 0% to 70% inactive
Features	Gathers per key	0 to 3 parent nodes gathered into the child table
Features	Summaries per key	1 to 2 of count, sum, mean, max, min, between static tables
Features	Per-table SCM	The single-table prior of Part 1: 3 to 16 nodes, 8 edge families, 14 marginals

- **Hierarchical:** each database first shifts the ranges themselves, and each table shifts them again. Databases differ as wholes, not only table by table.
- **Cost:** a median database has 29,000 rows and takes 0.29 s and 264 MiB to generate and save on a laptop. A corpus of 1,000 databases takes minutes.

PluRel API: A Single Table

The running example in PluRel 2. Each column declares its type, its marginal and its causes; **Hidden** declares a latent cause that no column records.

```
customers = Table({
    "id": PrimaryKey(),
    "age": Numeric(Uniform(18, 80), round=1),
    "income": Hidden(causes={"age": 1.0}),
    "engagement": Hidden(),
    "segment": Categorical({"bronze": 0.6, "silver": 0.3, "gold": 0.1},
            causes={"income": 1.0}),
    "visits": Numeric(Poisson(6), causes={"engagement": 1.0}),
    "spend": Numeric(LogNormal(3, 0.5), causes={"income": 1.0, "visits": 1.0}),
    "churn": Categorical({"no": 0.7, "yes": 0.3},
            causes={"engagement": -1.0, "spend": -1.0}),
})
frame = customers.sample(500, seed=0)
```

pip install "git+https://github.com/stanford-star/plurel.git@v2"

Further Key Structures

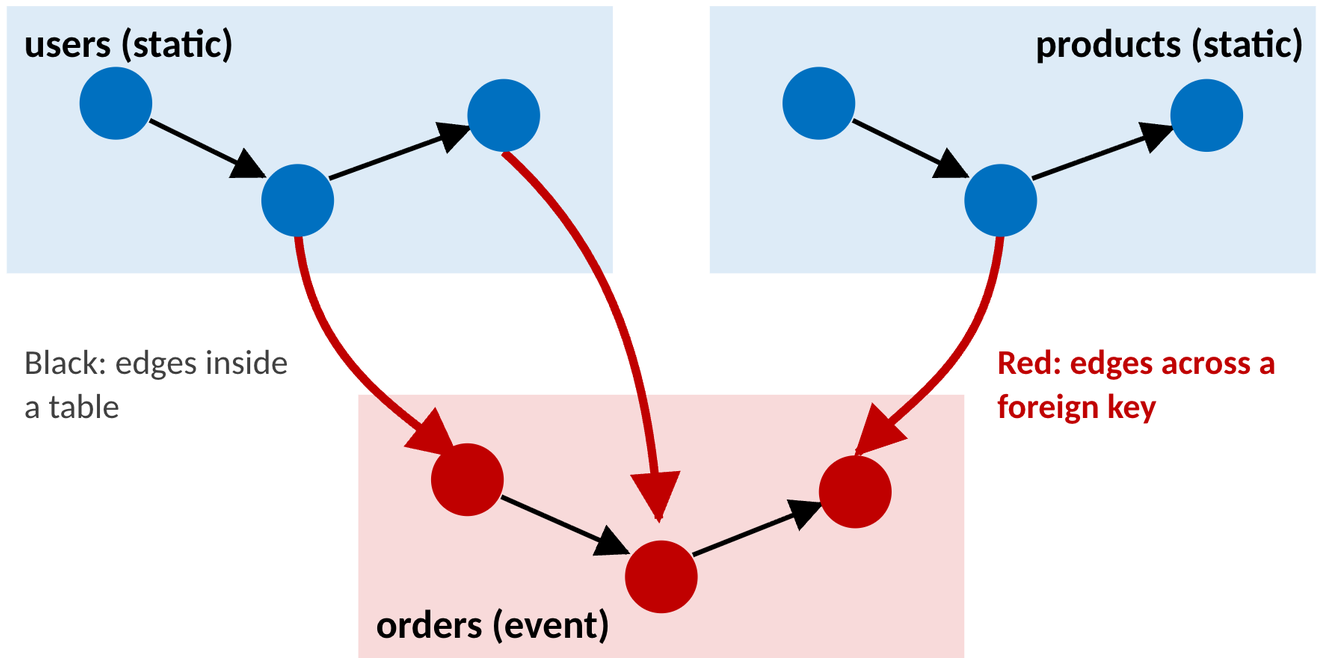
Real schemas contain a few more key patterns. The prior samples each of them:

Structure	How it is sampled	Real example
Uniform link	Parent row chosen uniformly at random	Random assignment
Self-reference	Row i of a table points to an earlier row j < i of the same table; a share of rows are roots. The rows form a forest.	Employee → manager; reply → comment
Nullable key	The key is blank with a fixed probability	Optional referrer
Two keys to one parent	The child holds two foreign keys into the same parent table	Sender and receiver; home and away team

- **In graph terms:** self-reference adds edges inside one node type; nullable keys remove edges; two keys to one parent create parallel edge types between the same pair of node types.
- A heterogeneous GNN must handle all three. A pretraining corpus without them leaves these cases untested.

One Global SCM over All Columns

The per-table SCMs and the crossing edges together form a **single directed acyclic graph** whose nodes are the columns of all tables.



- **Evaluation:** one pass in topological order over this graph. A database is a single ancestral sample, exactly as a table was in Part 1.
- **Rows are no longer i.i.d.:** two orders of the same customer share that customer's latents.
- **Acyclic by construction:** gathers go down a key and summaries go up, which could close a cycle. The prior wires summaries first, and a key never reads a node downstream of a summary.

Key idea: a relational database is one sample of one SCM. The keys decide which rows share ancestors.

Why SCMs as the Prior?

Alternative ways to generate a synthetic table, and what each can express:

Generator	Needs real data	Dependence it can express	Ground truth
Independent columns	No	None	Trivial
Gaussian copula	No	Pairwise, symmetric, monotone	Correlations only
Deep generative model (GAN, diffusion) fit to a real table	Yes: private, and one table at a time	Whatever that one table contains	Unknown
Random SCMs	No	Sparse, directed, nonlinear, mixed types	Fully known

- **Coverage:** confounders, mediators, colliders and hidden causes arise from random graphs without being designed in.
- **Control:** the noise level of every variable is a parameter, so tasks range from deterministic to unpredictable.
- **Evidence:** TabPFN and TabICL, pretrained only on SCM samples, match or beat tuned gradient-boosted trees on small real tables.

Foundation Models Reach Tabular and Relational Data

- Language and vision have foundation models: pretrain once, then use on new tasks without training. Since 2022 the same shift is happening for **tables**, and since 2025 for **relational databases**.

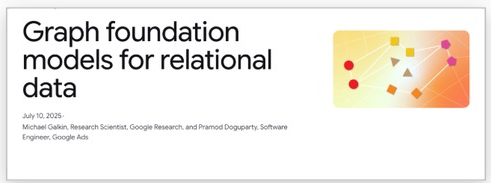

Article

Accurate predictions on small data with a tabular foundation model

https://doi.org/10.1038/s41586-024-08328-6 Noah Hollmann[1,2,3,7✉], Samuel Müller[1,7✉], Lennart Purucker[1], Arjun Krishnakumar[1], Max Körfer[1], Shi Bin Hoo[1], Robin Tibor Schirrmeister[4,5] & Frank Hutter[1,3,6✉]

2022–25: TabPFN (ICLR 2023, Nature 2025)

In 2.8 s, beats an ensemble of the strongest baselines tuned for 4 h, on tables of up to 10,000 rows. Pretrained only on synthetic tables.


Graph foundation models for relational data

July 10, 2025

Michael Galkin, Research Scientist, Google Research, and Pramod Doguparty, Software Engineer, Google Ads

2025: Google graph foundation model, KumoRFM

In-context prediction on relational databases. Google reports 3x to 40x gains in average precision on internal tasks such as ad spam.


RELATIONAL TRANSFORMER: TOWARD ZERO-SHOT FOUNDATION MODELS FOR RELATIONAL DATA

Rishabh Ranjan[0*], Valter Hudovernik[1†], Mark Znidar[2†], Charilaos Kanatsoulis[0], Roshan Upendra[3], Mahmoud Mohammadi[3], Joe Meyer[3], Tom Palczewski[3], Carlos Guestrin[0], Jure Leskovec[0]

[0]Stanford University, [1]Kumo AI, [2]University of Oxford, [3]SAP Labs LLC

2026: Relational Transformer (ICLR)

Zero-shot on unseen databases: 93% of supervised AUROC with 22M parameters, against 84% for a 27B-parameter LLM.


PLUREL: Synthetic Data unlocks Scaling Laws for Relational Foundation Models

Vignesh Kothapalli [1] Rishabh Ranjan [1] Valter Hudovernik [2] Vijay Prakash Dwivedi [1] Johannes Hoffart [3] Carlos Guestrin [1] Jure Leskovec [1]

2026: PluRel (ICML)

Synthetic relational databases for pretraining: this lecture.

- **Note:** TabPFN was pretrained only on **synthetic data**. PluRel brings that idea to relational databases.

The Rank Map: Definition and Properties

Definition. Given latent values $z_1, \ldots, z_n$ of a node and a target marginal F: draw $u_1, \ldots, u_n \sim F$ and sort them. The row with the k-th smallest latent receives the k-th smallest draw:

$$x_i = u_(\ \text{rank}(z_i)\ )$$

Example: n = 5, F = LogNormal(3, 0.5)

row	latent z	rank	column x
1	0.31	4	26.1
2	-1.20	1	11.2
3	2.05	5	41.7
4	0.02	3	20.3
5	-0.44	2	16.9

Sorted draws from F: 11.2, 16.9, 20.3, 26.1, 41.7
(illustrative values)

- **Monotone:** the order of rows is unchanged, so rank correlations between columns (the copula) are preserved.
- **Exact marginal:** the column is a sample from F, whatever the latent's shape. F can be log-normal, Pareto, a count distribution, or zero-inflated.
- **Categorical columns:** cut the latent at its quantiles to hit the class shares exactly, or take the argmax of class scores.
- **Separation of concerns:** the SCM decides dependence; the rank map decides marginals.

Modelling the Imperfections of Real Tables

A model pretrained on clean Gaussian columns fails on real ones. Each imperfection is a mechanism in the generator:

Phenomenon	Mechanism	Default prior
Unobserved variables	Nodes that no column records; they act as latent confounders	3 to 16 nodes, 3 to 12 columns
Missing at random	Each cell of a column is blank with a fixed probability	Rate 1% to 10%, on 0% to 60% of columns
Structured missingness	A cell is blank where a hidden two-class indicator node says so; the indicator depends on other variables	0% to 100% of the columns with missing values
Zero inflation	Marginal is a mixture: a point mass at 0 and an exponential	30% or 70% zeros
Outliers	Marginal $0.97 \cdot N(0, 1) + 0.03 \cdot N(0, 8^2)$	One of 14 marginal choices
Discretisation	A numeric node is recorded as 2 to 10 ordered bins	0% to 40% of numeric columns
Non-Gaussian noise	ε is Laplace, Student-t(4) or a centred Gamma	3 of 7 units of weight

- **Example draw** (table prior, seed 1): 12 categorical columns, 8 of them with 9% to 12% missing cells.