Second MODE Workshop on Differentiable Programming for Experiment Design
13 September 2022, 18h20, Kolymbari, Greece.

Differentiable Programming on relational languages

Paul Peseux (Lokad & LITIS)

Differentiable Programming on relational languages

Problem:

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(X)$$

Problem:

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(X)$$

Differentiable programming:

Direct access to $\nabla_\theta f = \begin{pmatrix} \frac{\partial f}{\partial \theta_1} \\ \frac{\partial f}{\partial \theta_2} \\ \dots \\ \frac{\partial f}{\partial \theta_p} \end{pmatrix}$

Problem:

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(X)$$

Differentiable programming:

Direct access to $\nabla_\theta f = \begin{pmatrix} \frac{\partial f}{\partial \theta_1} \\ \frac{\partial f}{\partial \theta_2} \\ \dots \\ \frac{\partial f}{\partial \theta_p} \end{pmatrix}$

Gradient descent algorithms

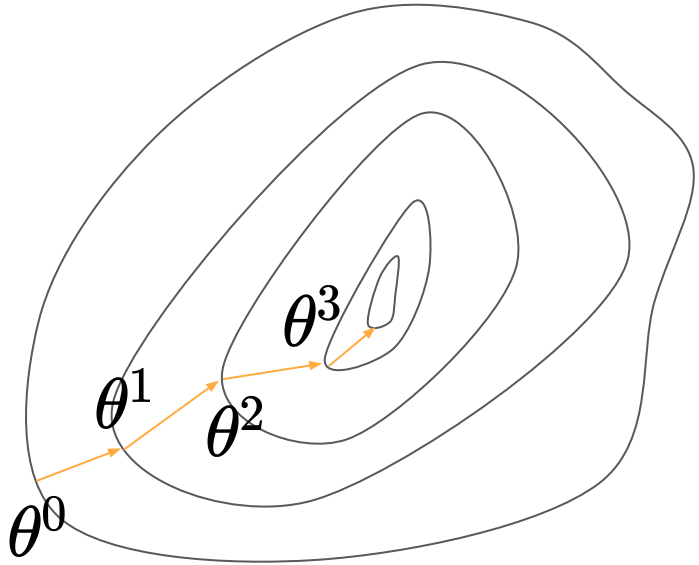


Fig.1 Gradient Descent

Differentiable Programming on relational languages

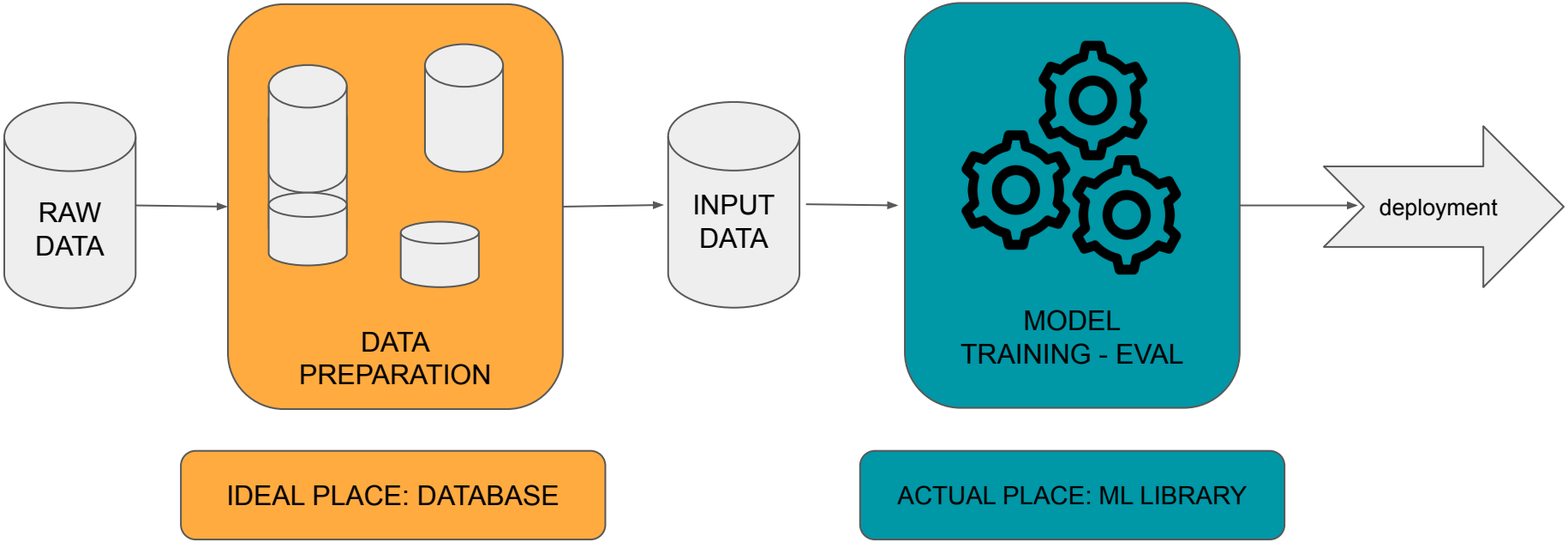


Fig.2 Classical pipeline

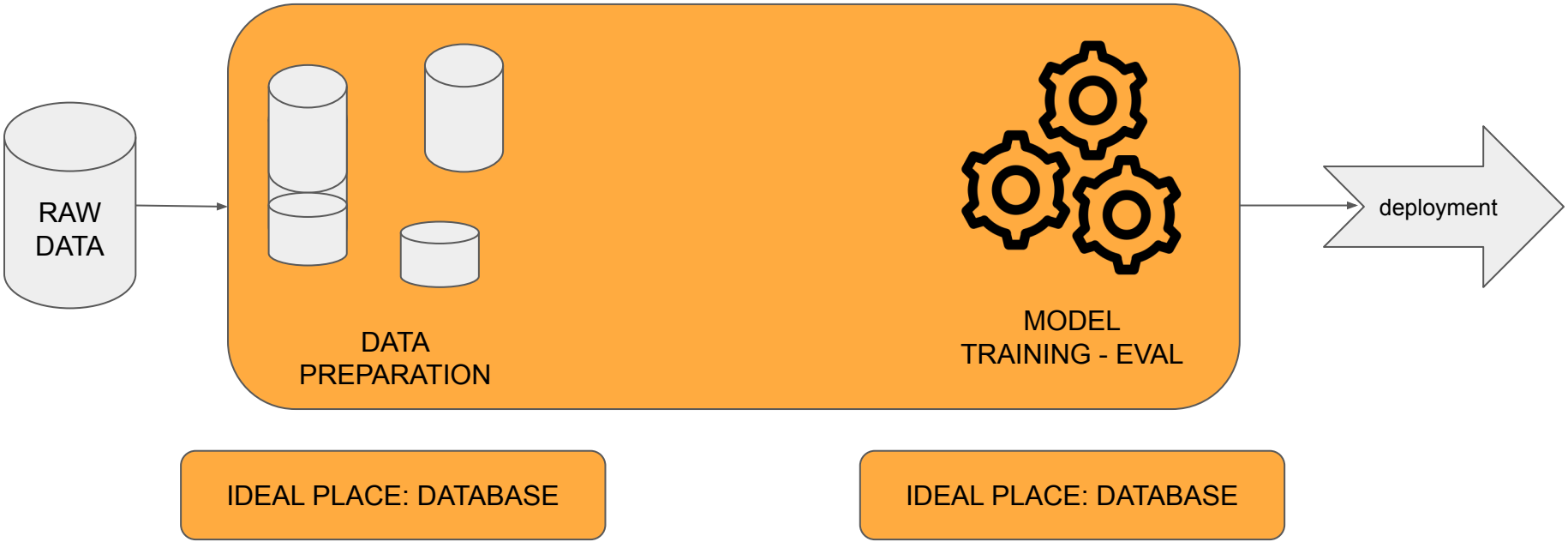
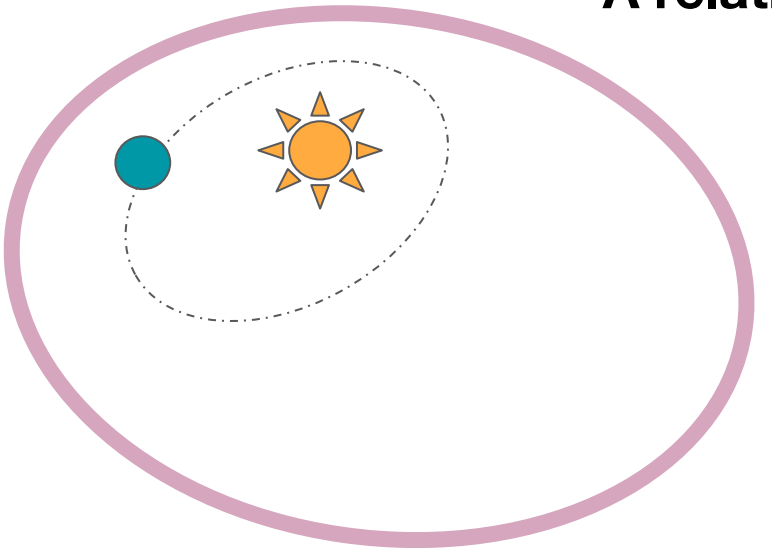


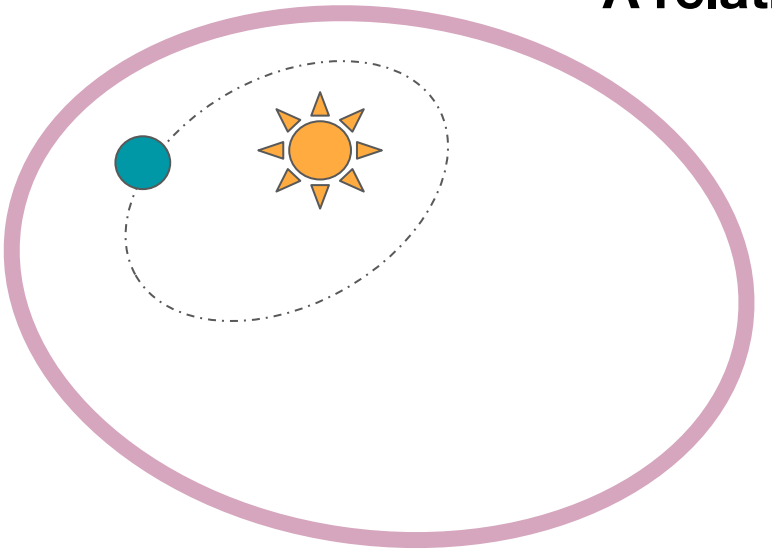
Fig.3 Proposed pipeline

Fig.3 Proposed pipeline

A relational example:

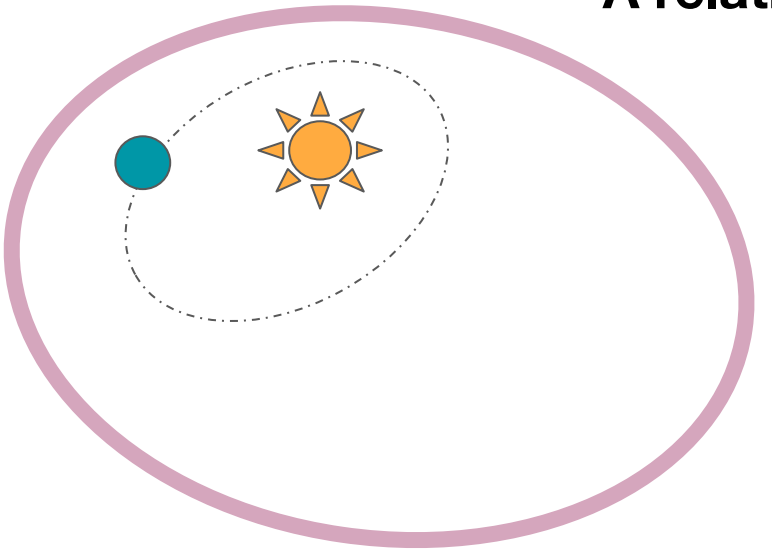


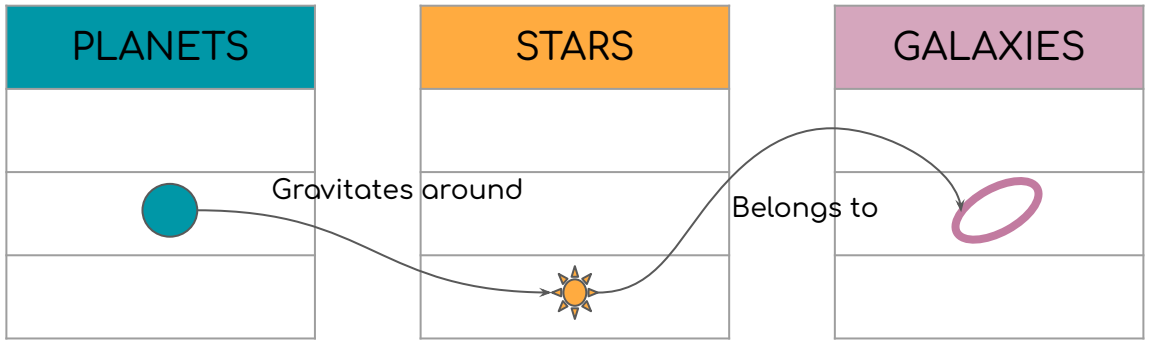
A relational example:



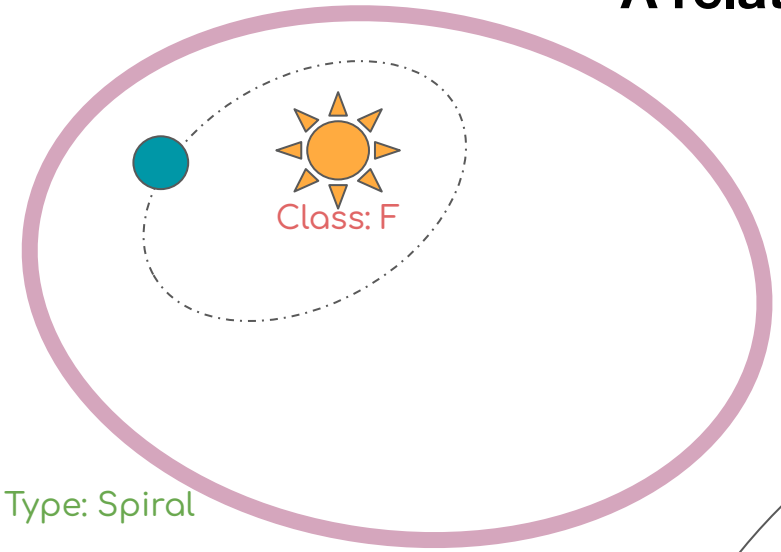
PLANETS	STARS	GALAXIES
		
		

A relational example:





A relational example:



CLASSES
O
B
A
F
G
K
...

Is of class

PLANETS	STARS	GALAXIES

Gravitates around

Belongs to

Is of type

TYPES
Spiral
Elliptical
Lenticular
...

A relational example:

Data stored in relational databases

- Oracle Database
- MySQL
- Microsoft SQL Server
- PostgreSQL
- IBM Db2
- Microsoft Access
- SQLite
- MariaDB
- Snowflake
- Microsoft Azure SQL Database
- Apache Hive
- Teradata Vantage
- …

Some are queryable online

- Simbad
- Exoplanet.eu
- …

CLASSES
O
B
A
F
G
K
…

PLANETS


STARS


GALAXIES


TYPES
Spiral
Elliptical
Lenticular
…

PLANETS	STARS	GALAXIES	CLASSES	TYPES

Objective:

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(X)$$

Objective:

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(X)$$

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(x_1, x_2, \ldots, x_n)$$

Objective:

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(X)$$

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(x_1, x_2, \ldots, x_n)$$

Inputs come from the database

Objective:

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(X)$$

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(x_1, x_2, \ldots, x_n)$$

Inputs come from the database

Loss example:

$$f(planet) = | \ age(star(planet)) * \mu_{type(galaxy(planet))} * \gamma_{class(star(planet))} - weight(planet)|^2$$

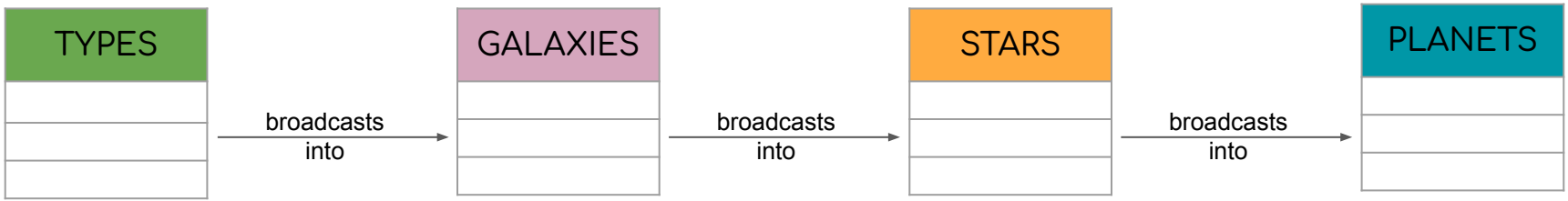
Objective:

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(X)$$

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(x_1, x_2, \ldots, x_n)$$

f is a QUERY

f is not *just* a mathematical function, it is also *relational*



Objective:

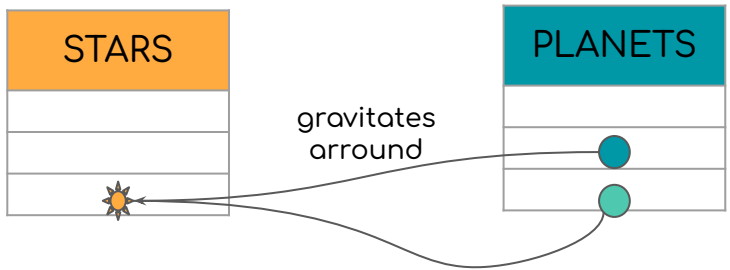
$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(X)$$

$$\theta^* = \underset{\theta \in \Theta}{argmin} \sum_{X \in D} f_\theta(x_1, x_2, \ldots, x_n)$$

f is a QUERY

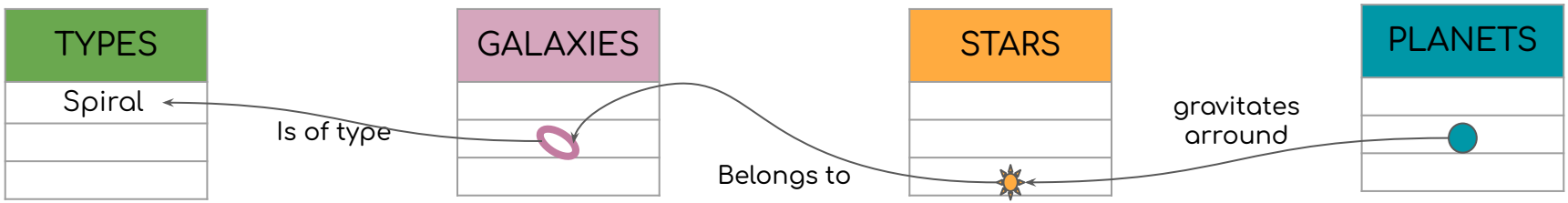
f is not *just* a mathematical function, it is also *relational*

broadcasts: for each star, one can access all its planets



f is a QUERY

f is not *just* a mathematical function, it is also *relational*

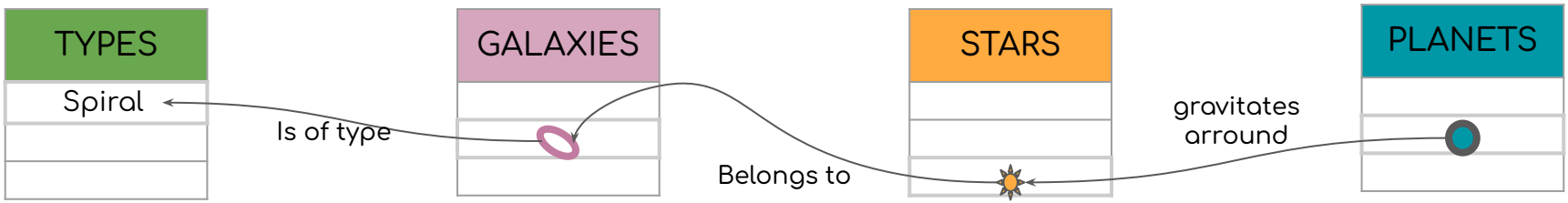


Learn parameters per

- star
- galaxy
- galaxy type …

f is a QUERY

f is not *just* a mathematical function, it is also *relational*



Piece of SQL code

```
SELECT P.planet, f(P.x1, S.x2, G.x3) as loss
    FROM (SELECT P.planet, P.star, P.x1 FROM PLANETS P)
    TOTAL JOIN
        (SELECT S.star S.class, S.galaxy, G.type, S.x2, G.x3
          FROM (SELECT S.star, S.class, S.galaxy FROM STARS S)
          TOTAL JOIN
            (SELECT G.galaxy, G.type FROM GALAXIES G)
            (WHERE S.galaxy = G.galaxy))
        WHERE P.star = S.star
```

f is a QUERY

f is not *just* a mathematical function, it is also *relational*

Piece of SQL code

```
SELECT P.planet, f(P.x1, S.x2, G.x3) as loss
    FROM (SELECT P.planet, P.star, P.x1 FROM PLANETS P)
    TOTAL JOIN
        (SELECT S.star S.class, S.galaxy, G.type, S.x2, G.x3
          FROM (SELECT S.star, S.class, S.galaxy FROM STARS S)
          TOTAL JOIN
            (SELECT G.galaxy, G.type FROM GALAXIES G)
            (WHERE S.galaxy = G.galaxy))
        WHERE P.star = S.star
```

f is a QUERY

f is not *just* a mathematical function, it is also *relational*

TOTAL JOIN operator

Piece of SQL code

```
SELECT P.planet, f(P.x1, S.x2, G.x3) as loss
    FROM (SELECT P.planet, P.star, P.x1 FROM PLANETS P)
    TOTAL JOIN
        (SELECT S.star S.class, S.galaxy, G.type, S.x2, G.x3
          FROM (SELECT S.star, S.class, S.galaxy FROM STARS S)
          TOTAL JOIN
            (SELECT G.galaxy, G.type FROM GALAXIES G)
            (WHERE S.galaxy = G.galaxy))
        WHERE P.star = S.star
```

TOTAL JOIN:

$T1$ TOTAL JOIN $T2$ on $\langle\theta\rangle$ is the same semantic as $T1$ INNER JOIN $T2$ on $\langle\theta\rangle$ with the additional constraint that for each line of $T1$, there is ***exactly*** one line of $T2$ that corresponds.

To make a successful $T1$ TOTAL JOIN $T2$ ON $\langle\theta\rangle$ it is sufficient that θ columns are a primary key in $T1$ and a foreign key in $T2$, but it is not necessary.

f is a QUERY

f is not *just* a mathematical function, it is also *relational*

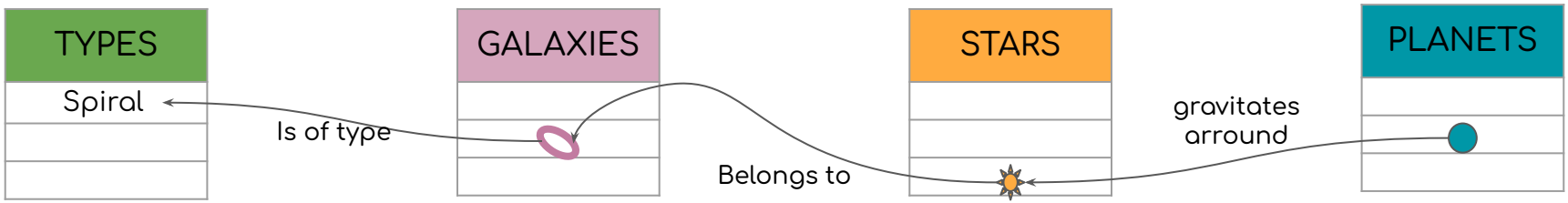
TOTAL JOIN operator

Piece of SQL code

```
SELECT P.planet, f(P.x1, S.x2, G.x3) as loss
    FROM (SELECT P.planet, P.star, P.x1 FROM PLANETS P)
    TOTAL JOIN
        (SELECT S.star S.class, S.galaxy, G.type, S.x2, G.x3
          FROM (SELECT S.star, S.class, S.galaxy FROM STARS S)
          TOTAL JOIN
            (SELECT G.galaxy, G.type FROM GALAXIES G)
            (WHERE S.galaxy = G.galaxy))
        WHERE P.star = S.star
```

f is a QUERY

f is not *just* a mathematical function, it is also *relational*



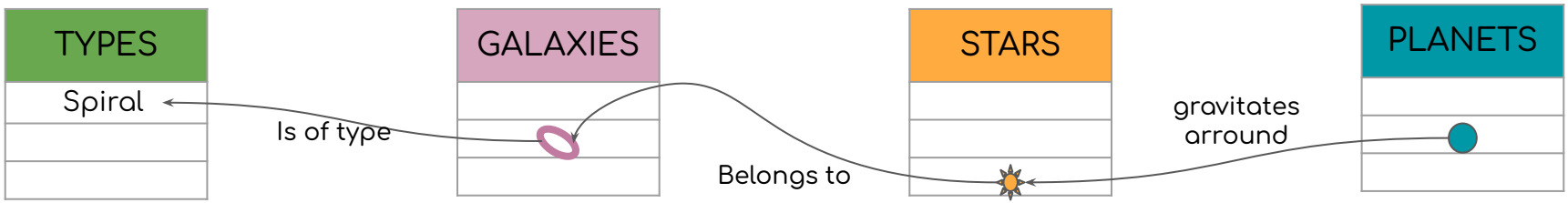
Piece of Envision code (Lokad DSL)

```
autodiff Planets with
      params Planets.theta1   auto
      params Stars.theta2     auto
      params Galaxies.theta3  auto
      params Types.theta4     auto

      return Mode.f(Planets.theta1, Stars.theta2, Galaxies.theta3, Types.theta4,
                    Planets.Weight, Stars.Weight, …)
```

f is a QUERY

f is not *just* a mathematical function, it is also *relational*



Piece of Envision code (Lokad DSL)

```
autodiff Planets with
      params Planets.theta1   auto
      params Stars.theta2     auto
      params Galaxies.theta3  auto
      params Types.theta4     auto

      return Mode.f(Planets.theta1, Stars.theta2, Galaxies.theta3, Types.theta4,
                    Planets.Weight, Stars.Weight, …)
```

f is a QUERY

f is not *just* a mathematical function, it is also *relational*

Piece of Envision code (Lokad DSL)

```
autodiff Planets with
      params Planets.theta1   auto
      params Stars.theta2     auto
      params Galaxies.theta3  auto
      params Types.theta4     auto

      return Mode.f(Planets.theta1, Stars.theta2, Galaxies.theta3, Types.theta4,
                    Planets.Weight, Stars.Weight, …)
```

f is a QUERY

f is not *just* a mathematical function, it is also *relational*

TOTAL JOIN operator

Piece of Envision code (Lokad DSL)

```
autodiff Planets with
    params Planets.theta1   auto
    params Stars.theta2     auto
    params Galaxies.theta3  auto
    params Types.theta4     auto

    return Mode.f(Planets.theta1, Stars.theta2, Galaxies.theta3, Types.theta4,
                  Planets.Weight, Stars.Weight, …)
```

Advantages:

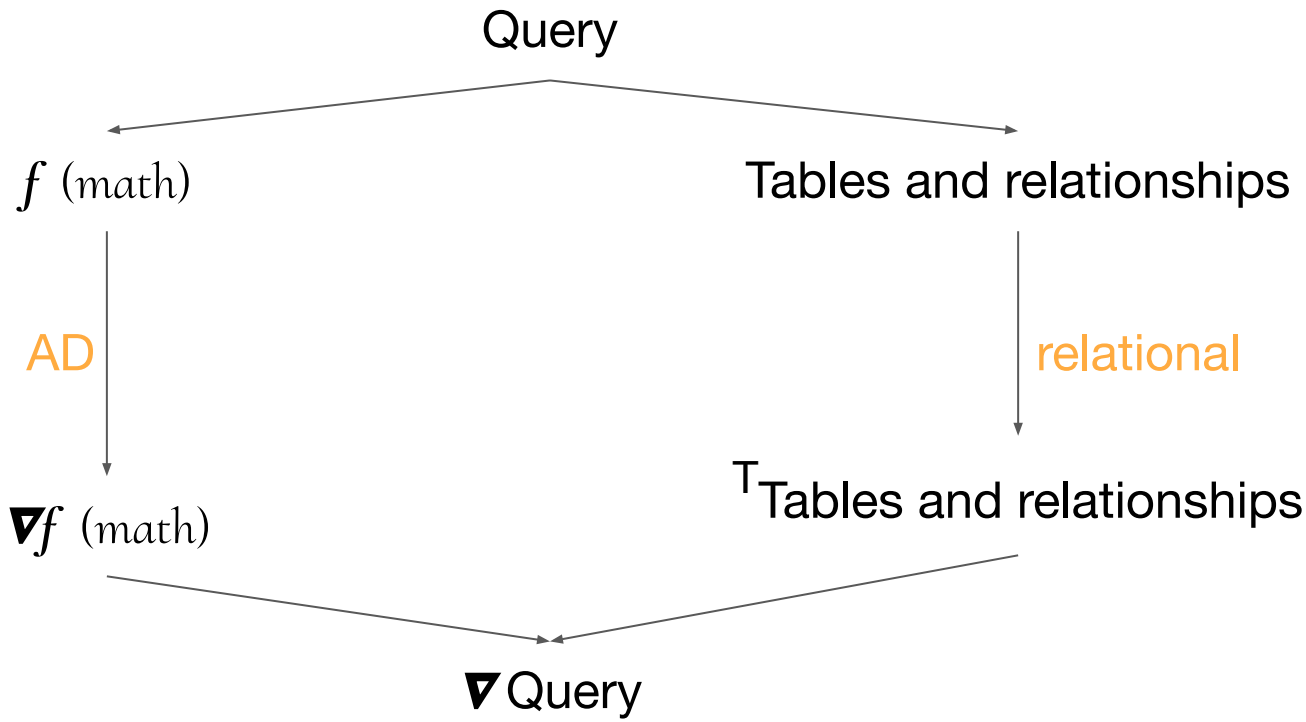
✅ expressiveness

Advantages:

✅ expressiveness

✅ preprocessing and machine learning at the same place

Advantages:


Query
f (math)
Tables and relationships
AD
relational
∇f (math)
$^{\mathsf{T}}$Tables and relationships
∇Query

Advantages:

✅ expressiveness

✅ preprocessing and machine learning at the same place

✅ memory consumption

Advantages:

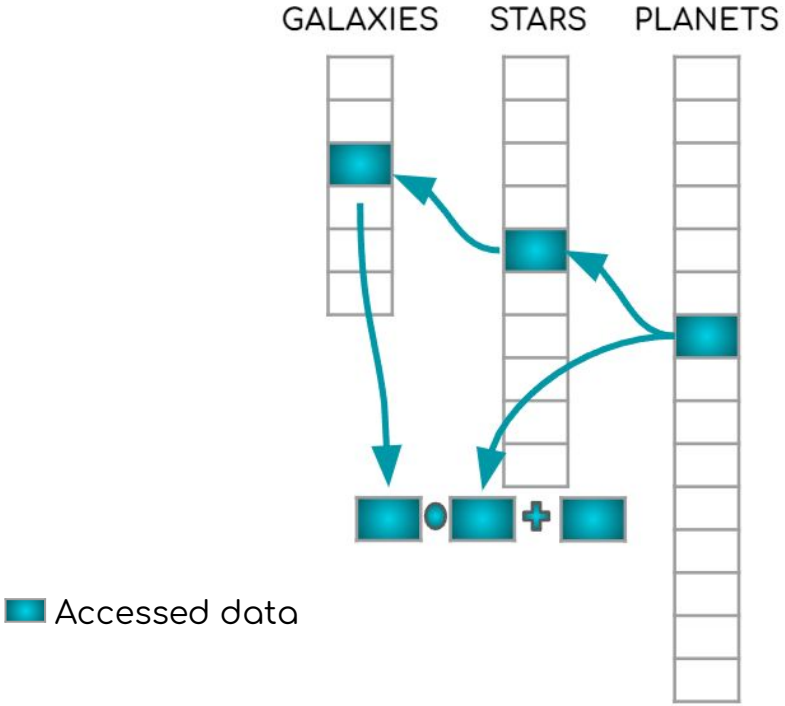


Fig.4 Relational approach

VS

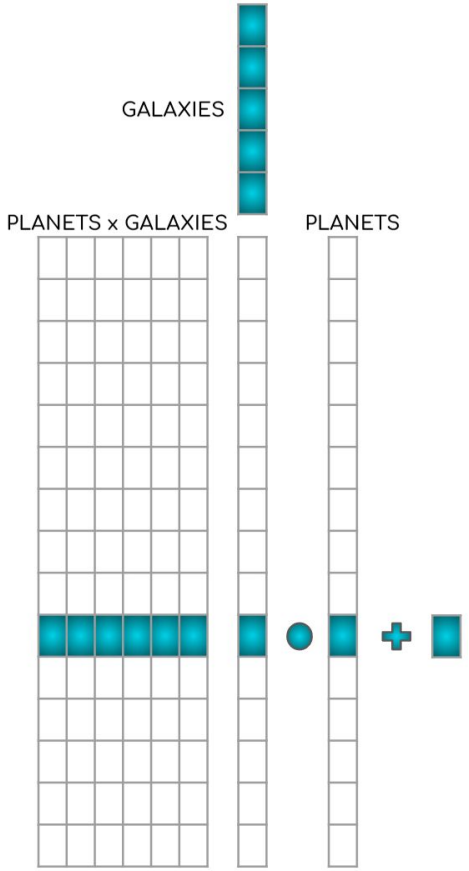


Fig.5 Matrix approach

Conclusion:

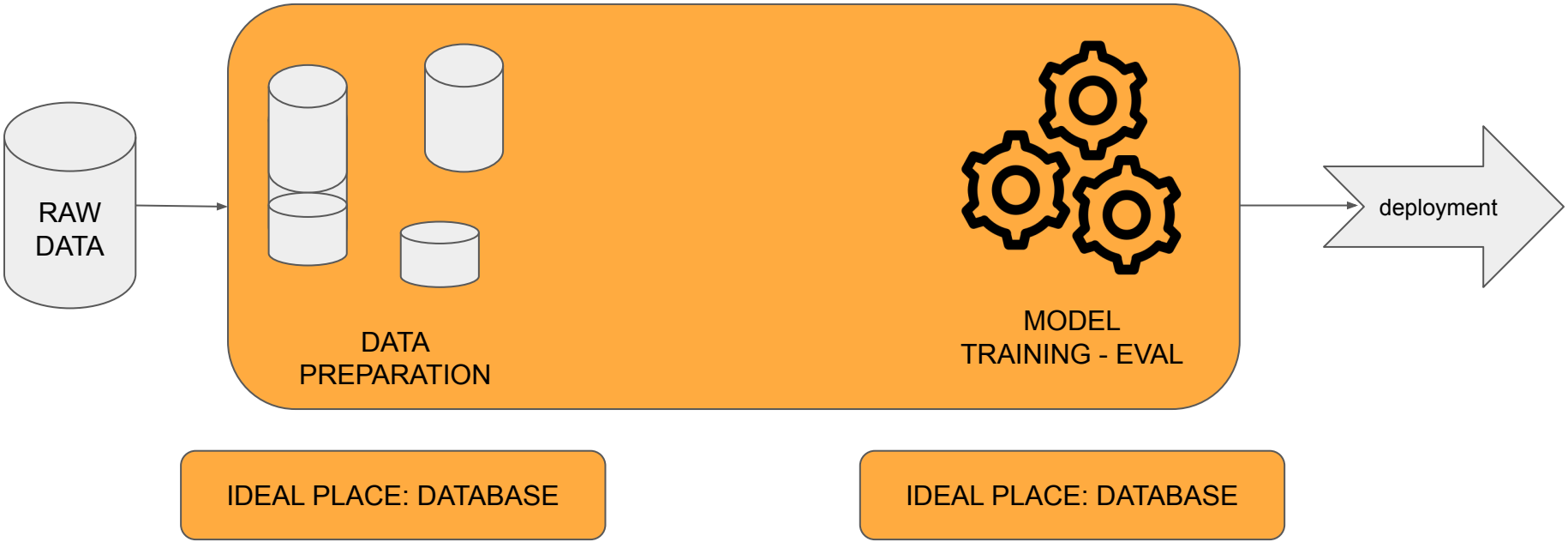


Fig.3 Proposed pipeline

Conclusion:

Differentiable Programming on relational languages

- developed on Envision (not open source yet)

- SQL : Schüle, Maximilian E., Frédéric Simonis, Thomas Heyenbrock, Alfons Kemper, Stephan Günnemann and Thomas Neumann. “In-Database Machine Learning: Gradient Descent and Tensor Algebra for Main Memory Database Systems.” BTW (2019).

Thank you

Any questions ?

contact: paul.peseux@gmail.com

References:

- Peseux, Paul. “Differentiating Relational Queries.” PhD@VLDB (2021).

- Schüle, Maximilian E., Frédéric Simonis, Thomas Heyenbrock, Alfons Kemper, Stephan Günnemann and Thomas Neumann. “In-Database Machine Learning: Gradient Descent and Tensor Algebra for Main Memory Database Systems.” BTW (2019).

- Wenger, Marc, Françoise Ochsenbein, Daniel Egret, Pascal Dubois, François Bonnarel, S. B. Borde, Françoise Genova, Gérard Jasniewicz, Suzanne Laloe, Soizick Lesteven and Richard Monier. “The SIMBAD astronomical database. The CDS reference database for astronomical objects.” Astronomy & Astrophysics Supplement Series 143 (2000): 9-22.

Simbad:

Portal | Simbad | VizieR | Aladin | X-Match | Other | Help

m

other query modes : Identifier query | Coordinate query | Criteria query | Reference query | Basic query | Script submission | TAP | Output options | Help

Query : m

submit id

Number of rows : 110

Show 100 entries

Search:

N	Identifier	catalog id	Otype	ICRS (J2000) RA	ICRS (J2000) DEC	Mag U	Mag B	Mag V	Mag R	Mag I	Sp type	#ref 1850 - 2022
1	M 1	M 1	SNR	05 34 32	+22 00.8						~	5947
2	M 2	M 2	GlC	21 33 27.02	-00 49 23.7			6.25			~	969
3	M 3	M 3	GlC	13 42 11.62	+28 22 38.2			6.39			~	2380
4	M 4	M 4	GlC	16 23 35.22	-26 31 32.7						~	1767
5	M 5	M 5	GlC	15 18 33.22	+02 04 51.7		7.34	5.95			~	1883
6	NGC 6405	M 6	OpC	17 40 17	-32 14.5						~	185
7	NGC 6475	M 7	OpC	17 53 47	-34 50.5						~	357
8	M 8	M 8	HII	18 03 37	-24 23.2						~	587
9	M 9	M 9	GlC	17 19 11.78	-18 30 58.5		9.36	8.42			~	298
10	M 10	M 10	GlC	16 57 09.05	-04 06 01.1			4.98			~	747
11	NGC 6705	M 11	OpC	18 51 04	-06 16.3		6.32	5.8			~	387
12	M 12	M 12	GlC	16 47 14.18	-01 56 54.7		8.52	6.07			~	627
13	M 13	M 13	GlC	16 41 41.634	+36 27 40.75			5.8			~	2110
14	M 14	M 14	GlC	17 37 36.15	-03 14 45.3		9.55	5.73			~	351

source: https://simbad.cds.unistra.fr/simbad/sim-fid