Invisible Glue: Scalable Self-Tuning Multi-Stores

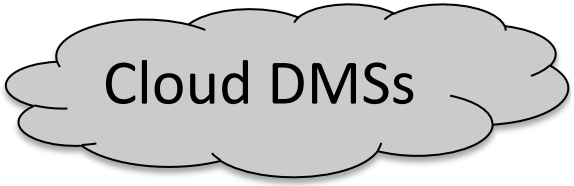
Francesca Bugiotti[§], Damian Bursztyn[§],
Alin Deutsch[¶],
Ioana Ileana[§], Ioana Manolescu[§]

[§] OAK team, INRIA, France
[¶] DB group, UC. California San Diego

The problem

- **Glut** of **varied** data management systems (DMS)
  - DM includes DBMS
- Different **data models**:
  - Relational, nested relational, tree, k-v, graphs, …

- Different **data access capabilities** (from simple API to various query languages)
- Different **architectures**: disk- vs. memory-based, centralized vs. distributed etc.
- Different **performance**
- Different levels of **transaction support**



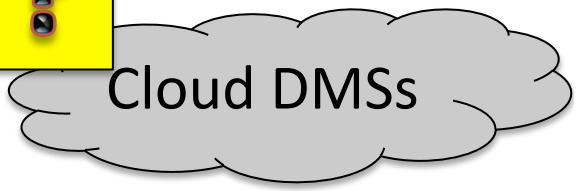
The problem

- **Glut** of **varied** data management systems (DMS)
  - DM includes DBMS
- Different **data models**:
  - Relation
- Different
- Different
- Different **performance**
- Different levels of **transaction support**

raphs, …

m simple

ory-based,



How do we get
performance
for a variety of datasets
on a variety of DMSs ?

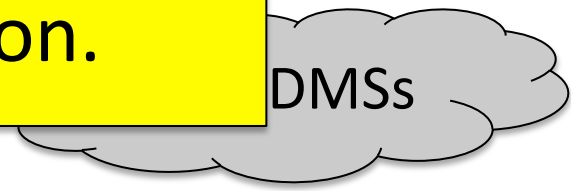
The problem

- **Glut** of **varied** data management systems (DMS)

How do we get **performance** for a vari… on a va…

…mple

- Differen… …based, centrali…



Focus not on **beating the most specialized optimizations** of the most specialized engine for a **given model**/application.

- Different **performance**
- Different levels of **transaction support**

The problem

- **Glut** of **varied** data management systems (DMS)

How do we get **performance** for a vari… on a va…

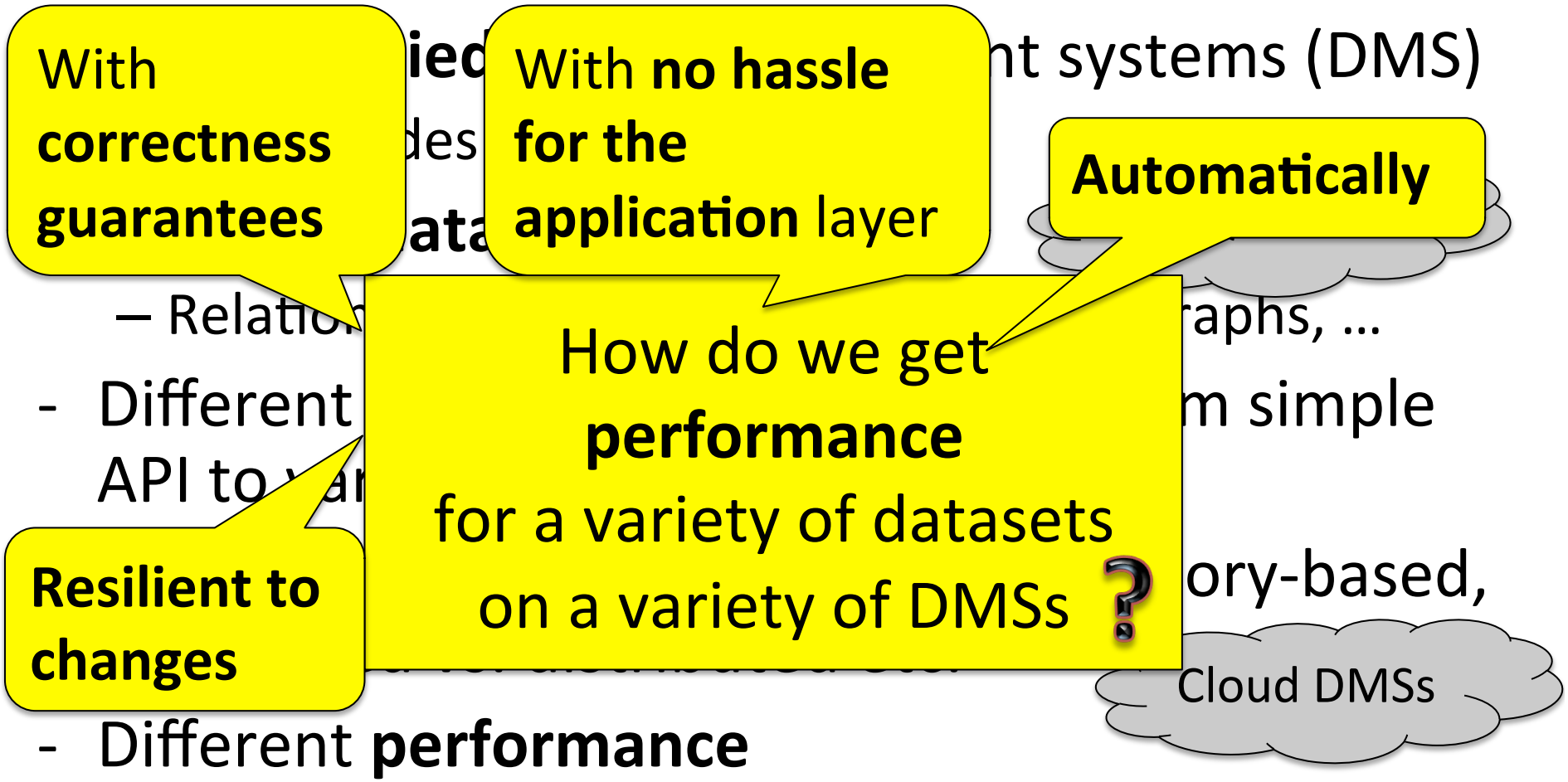
…mple

- Differen… centrali…
- Different **perfo…**
- Different levels…

Focus not on **beating the most specialized optimizations** of the most… giv…

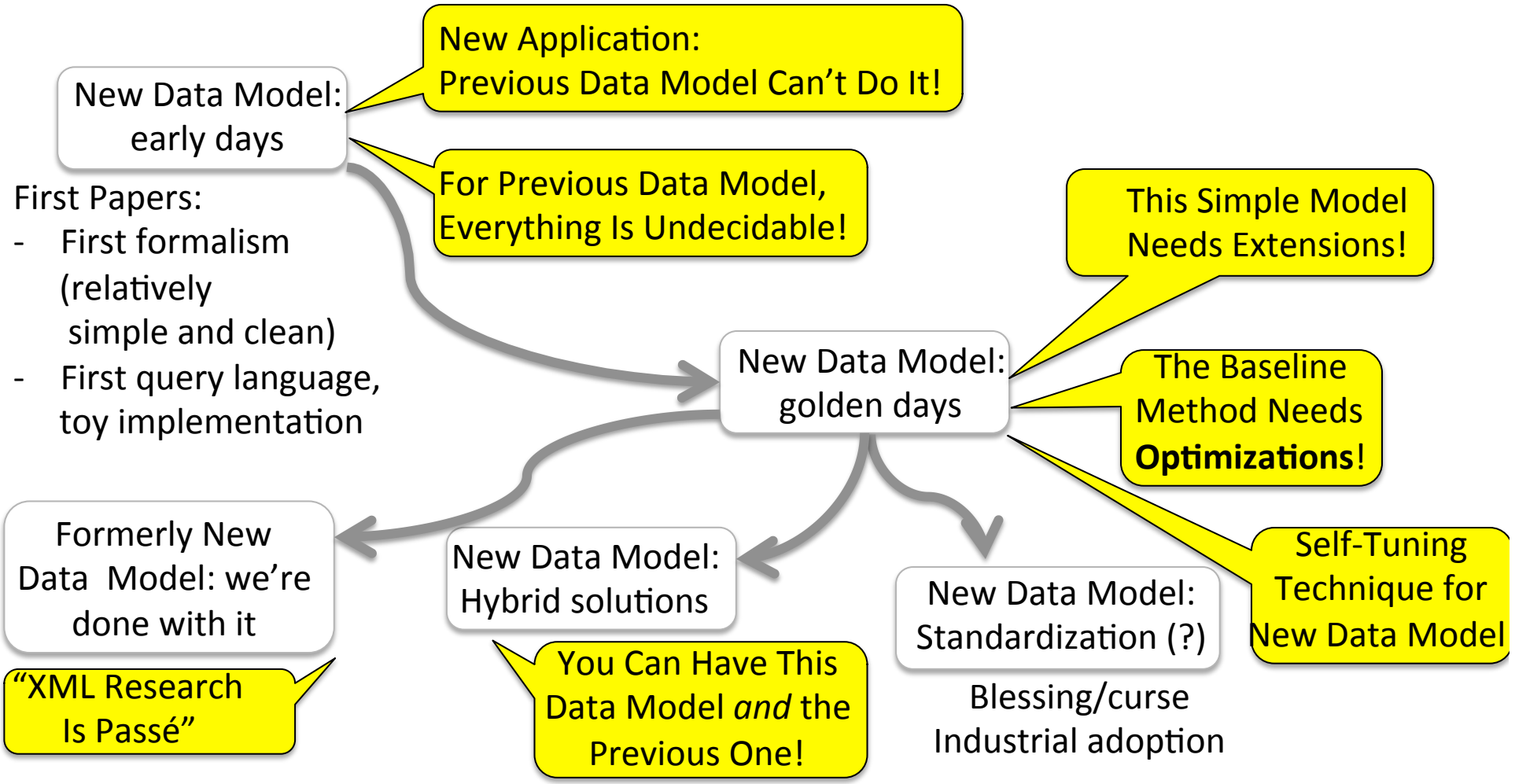
Focus on **robust performance for varied data models** across a **changing set of heterogeneous DMSs**

The problem, qualified



- Different **performance**
- Different levels of **transaction support**

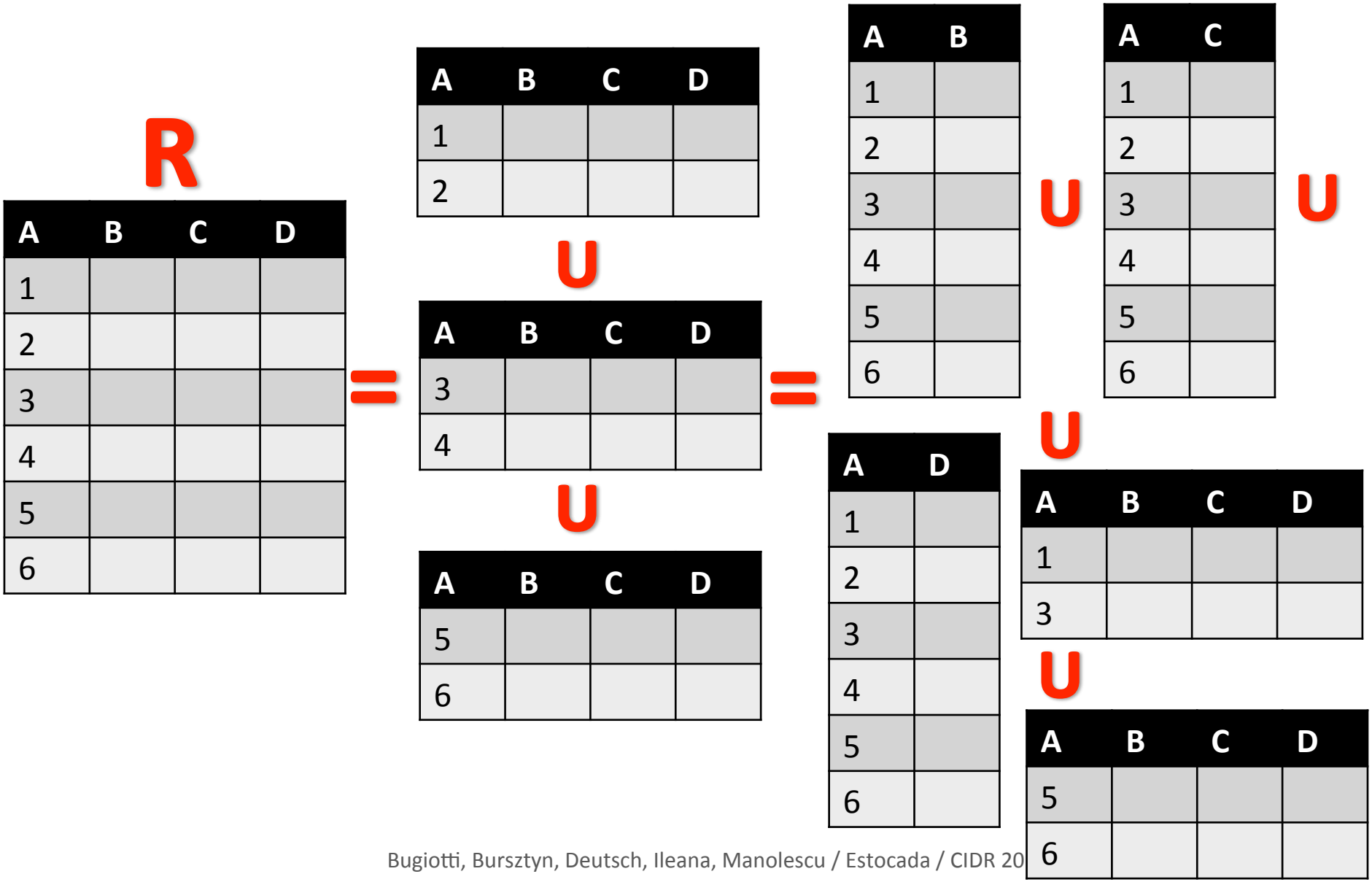
A piece of self-derision: The Next Data Model That Will Save The World



Estocada: invisible glue for heterogeneous stores

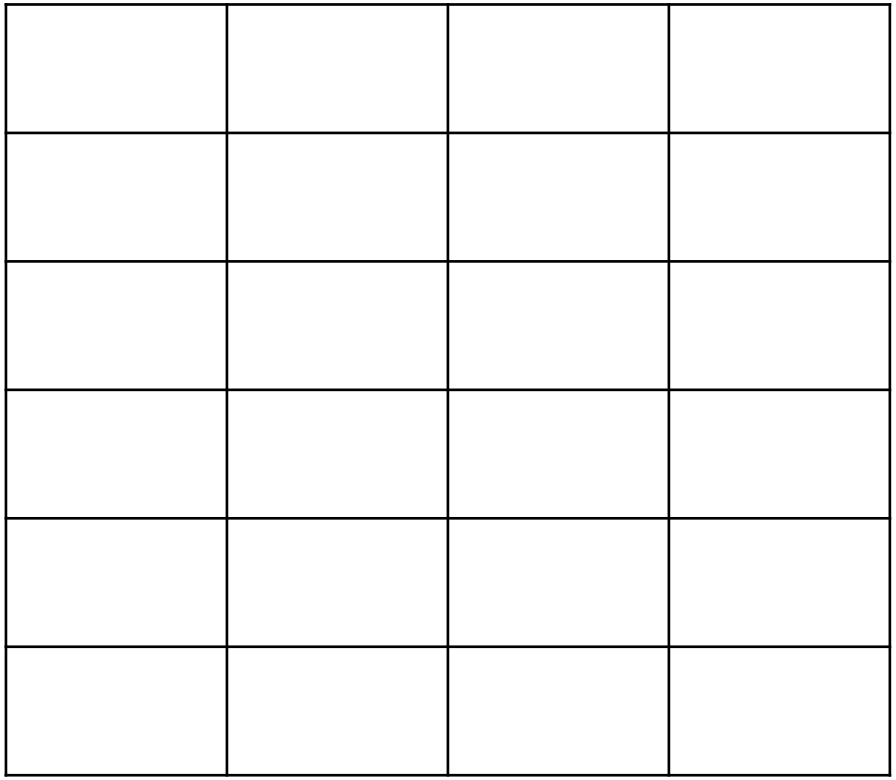
- Data models: **any/many** (side by side)
  - As the data is
- Systems: **any/many** (side by side)
  - Those available
- Store each data set as a set of **fragments**
  - Or splits / shards / partitions / indexes / materialized (potentially indexed) **views**
  - Each fragment resides in a DMS

Dataset fragmentations



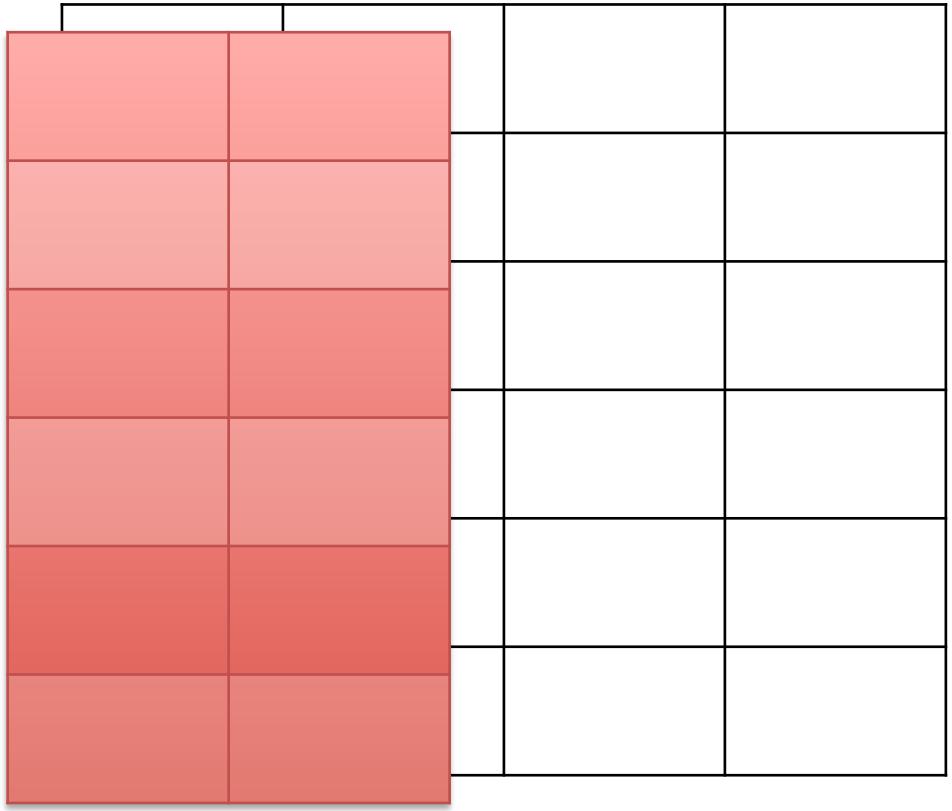
Dataset fragmentations

Example: relational dataset R



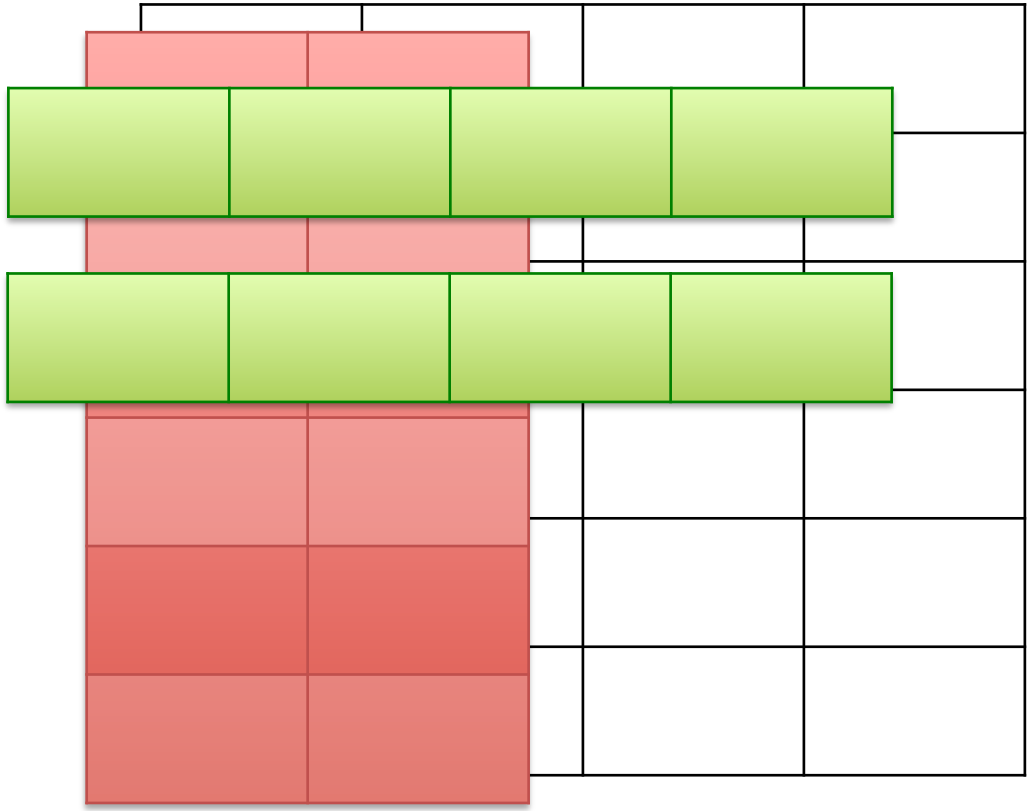
Dataset fragmentations

Example: relational dataset R



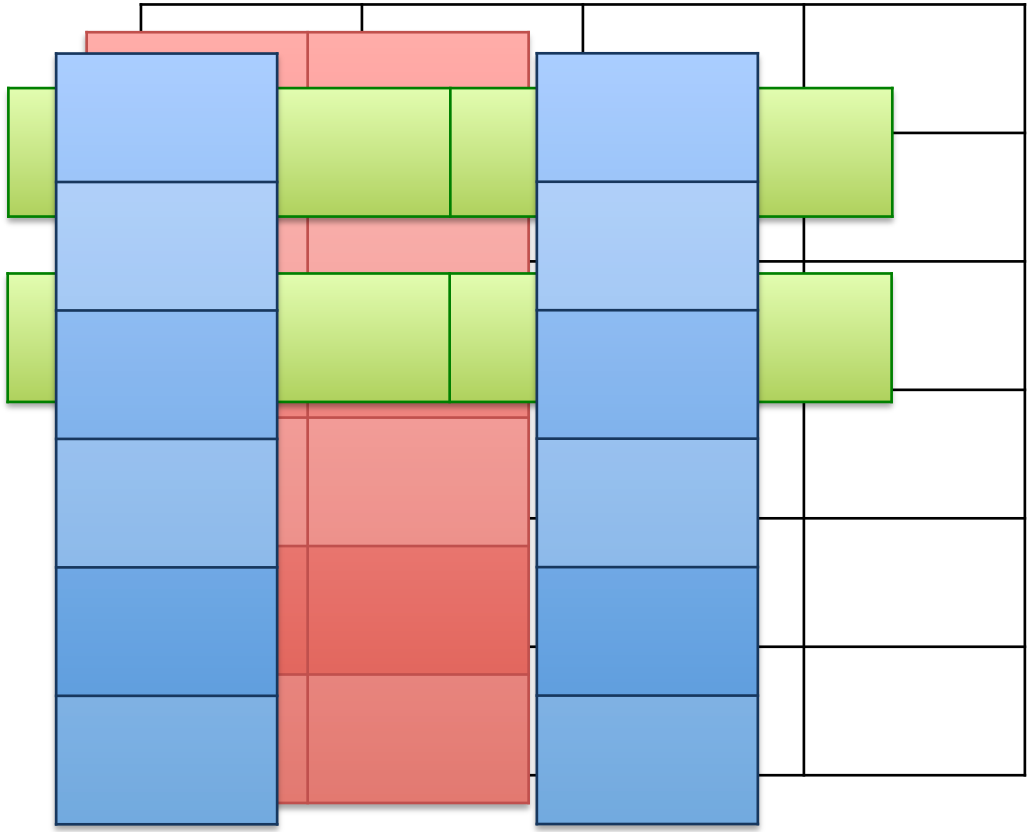
Dataset fragmentations

Example: relational dataset R



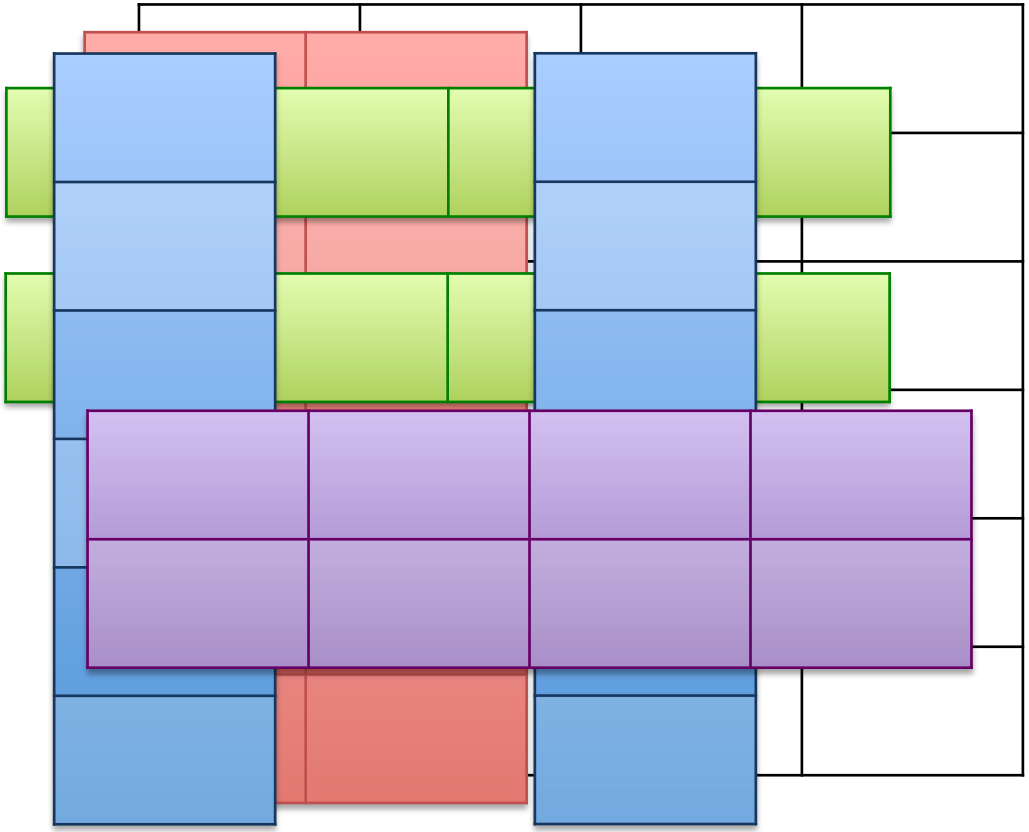
Dataset fragmentations

Example: relational dataset R



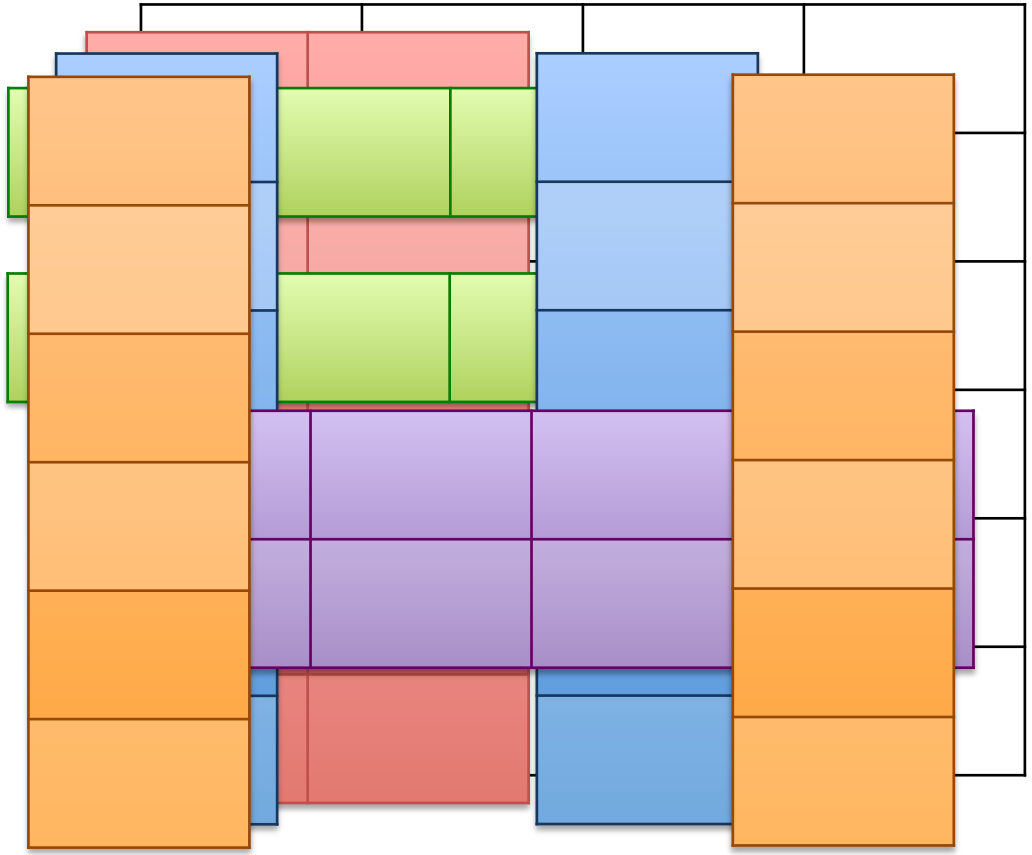
Dataset fragmentations

Example: relational dataset R



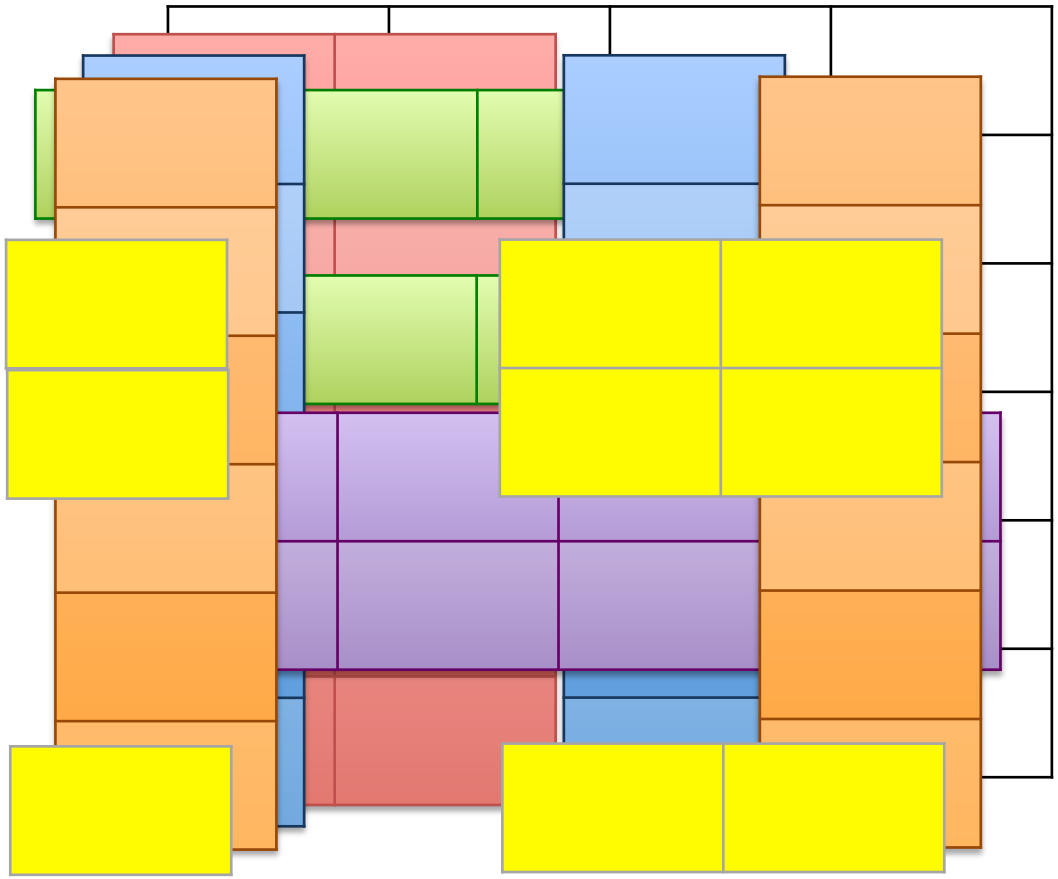
Dataset fragmentations

Example: relational dataset R



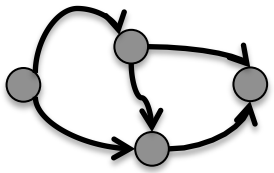
Dataset fragmentations

Example: relational dataset R



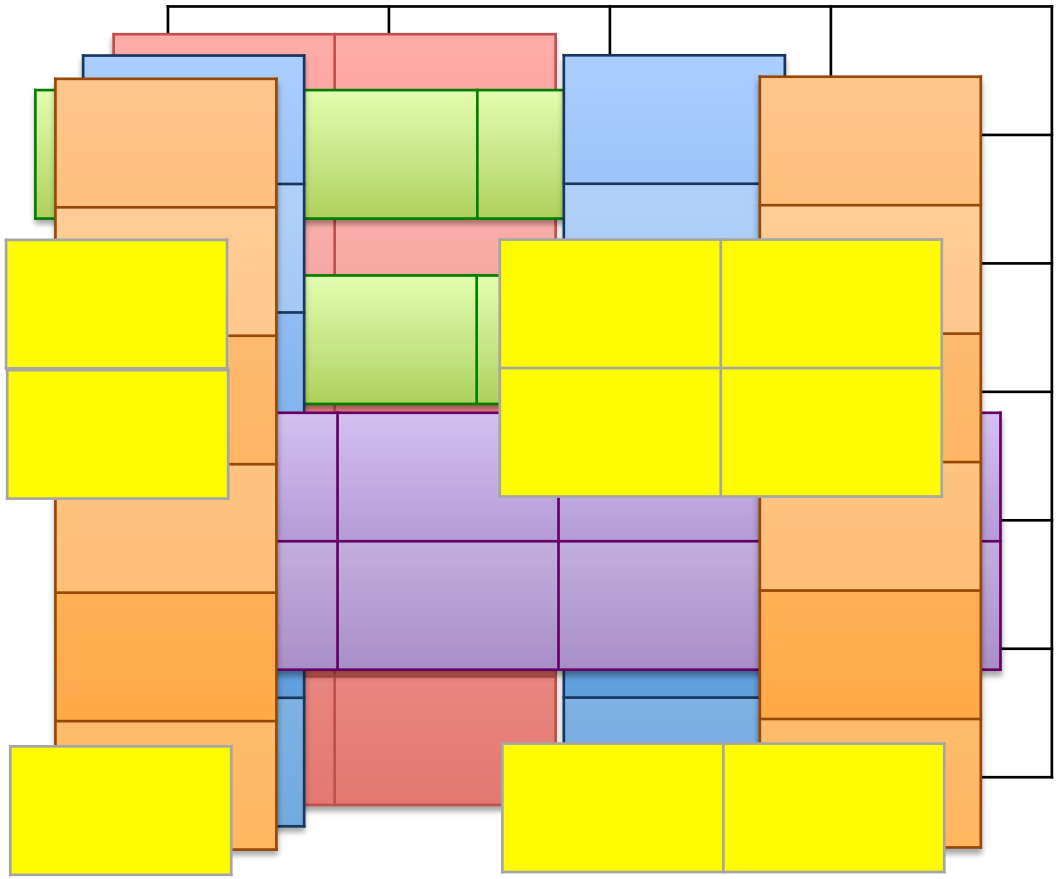
Fragmentations made of views

- The content of each fragment is described **declaratively**
- **Fragment = (materialized) view** [+ parameters]
  - « The name and addresses of all clients »
  - « The sales partitioned by zipcode »
- **Index = view with binding pattern**
  - « The name and addresses of all clients, by their age and zipcode »
  - Also: navigation in trees or graphs key-value stores

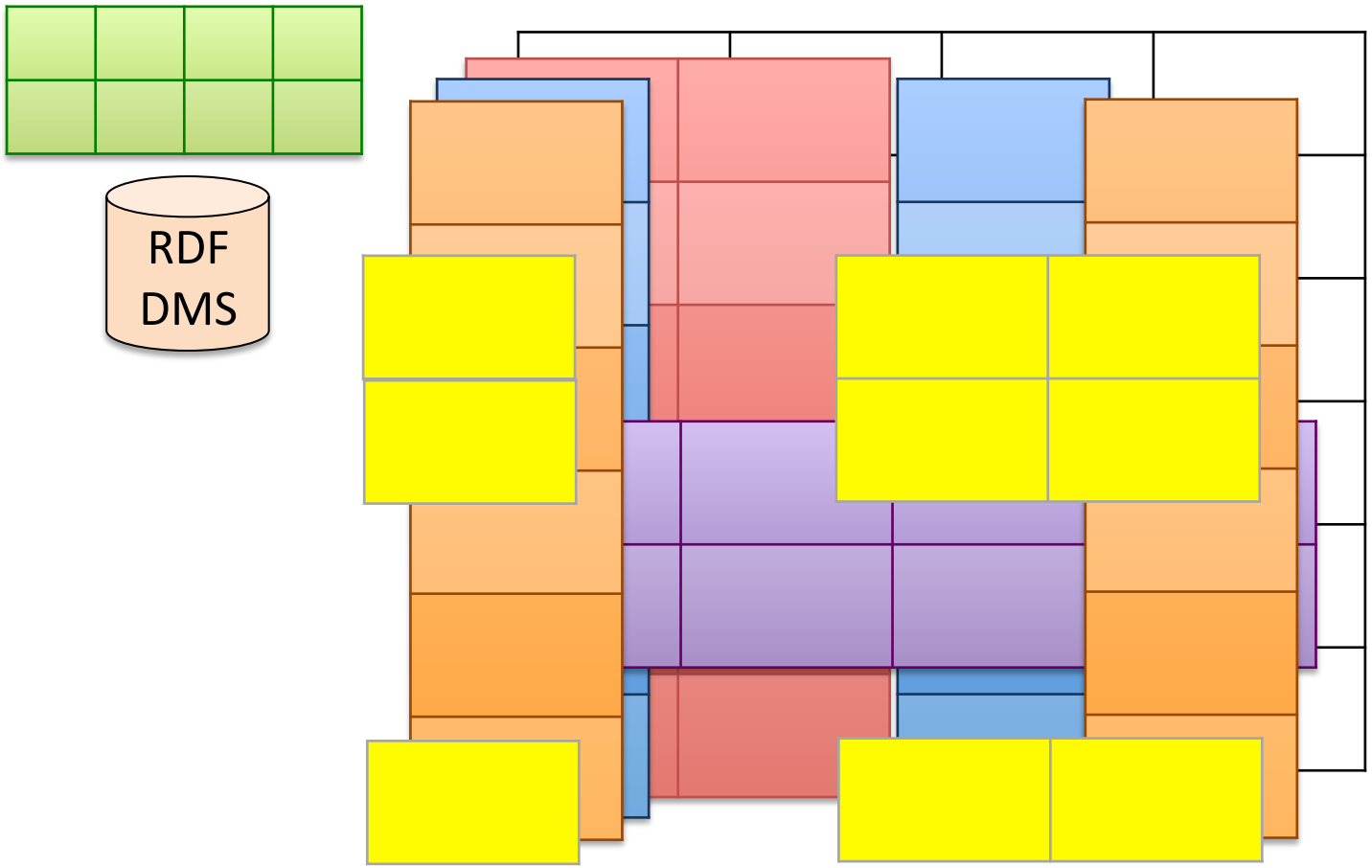


Fragment = materialized view [+ parameters] [+ binding pattern]

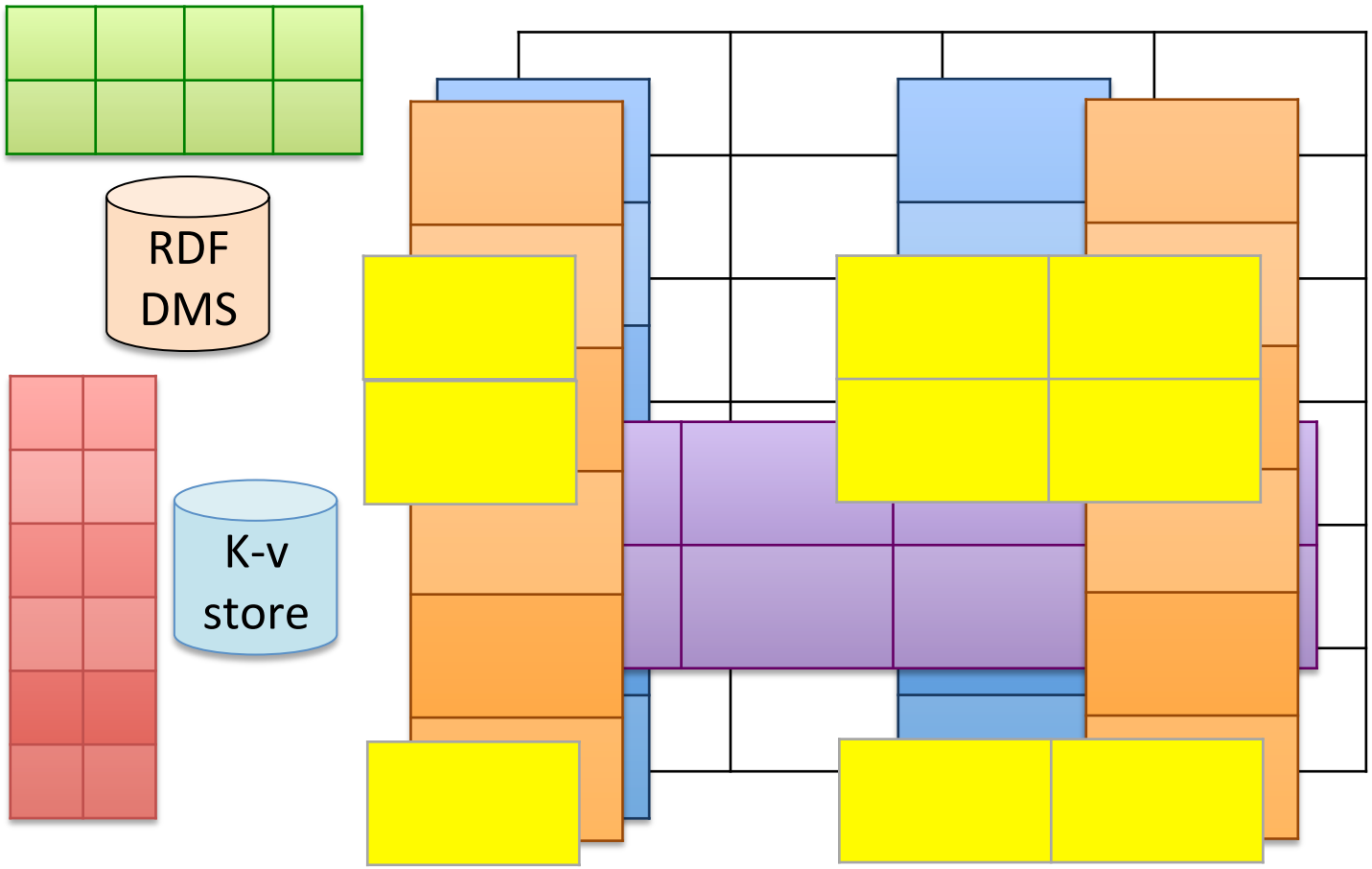
Fragments distribution across stores



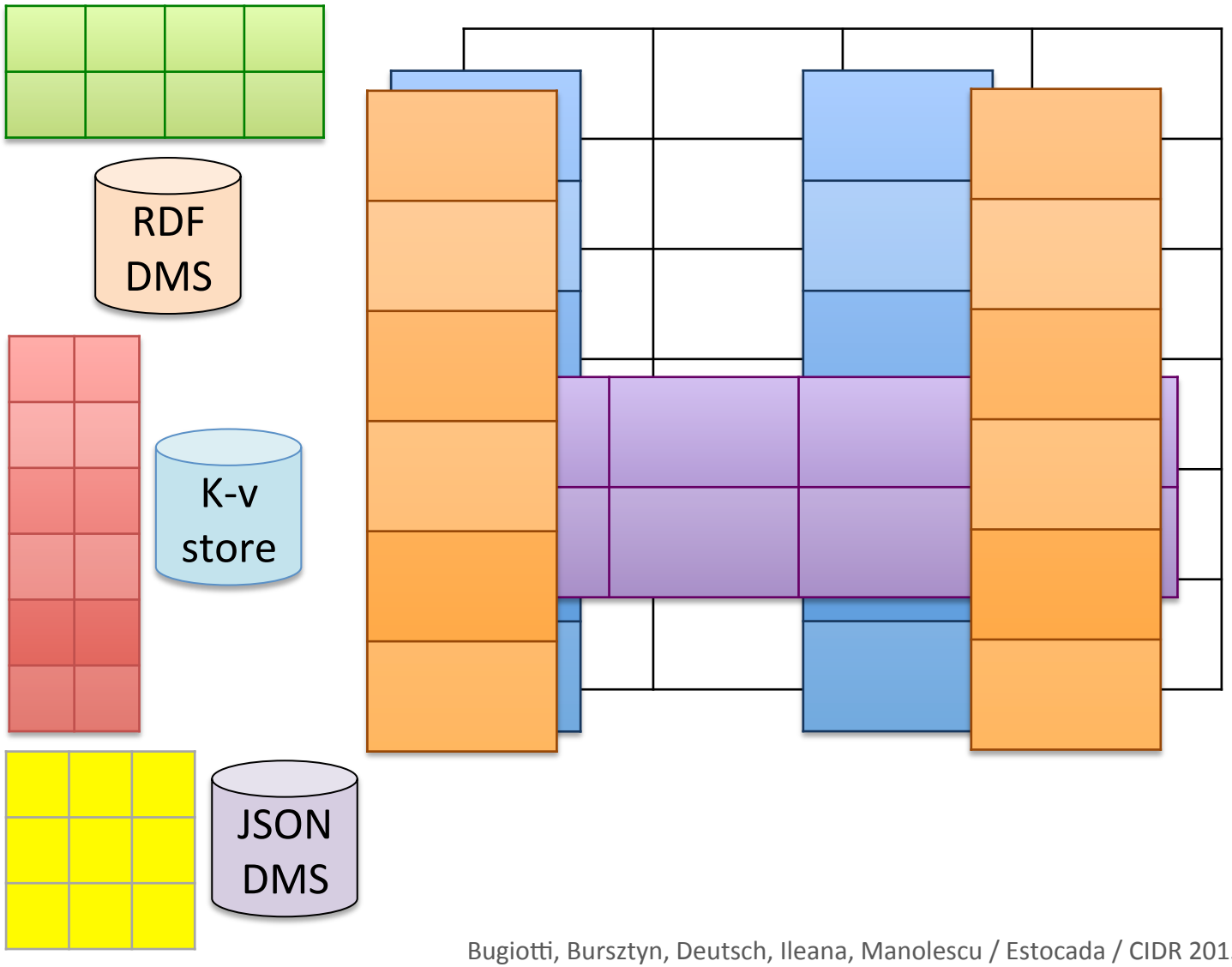
Fragments distribution across stores



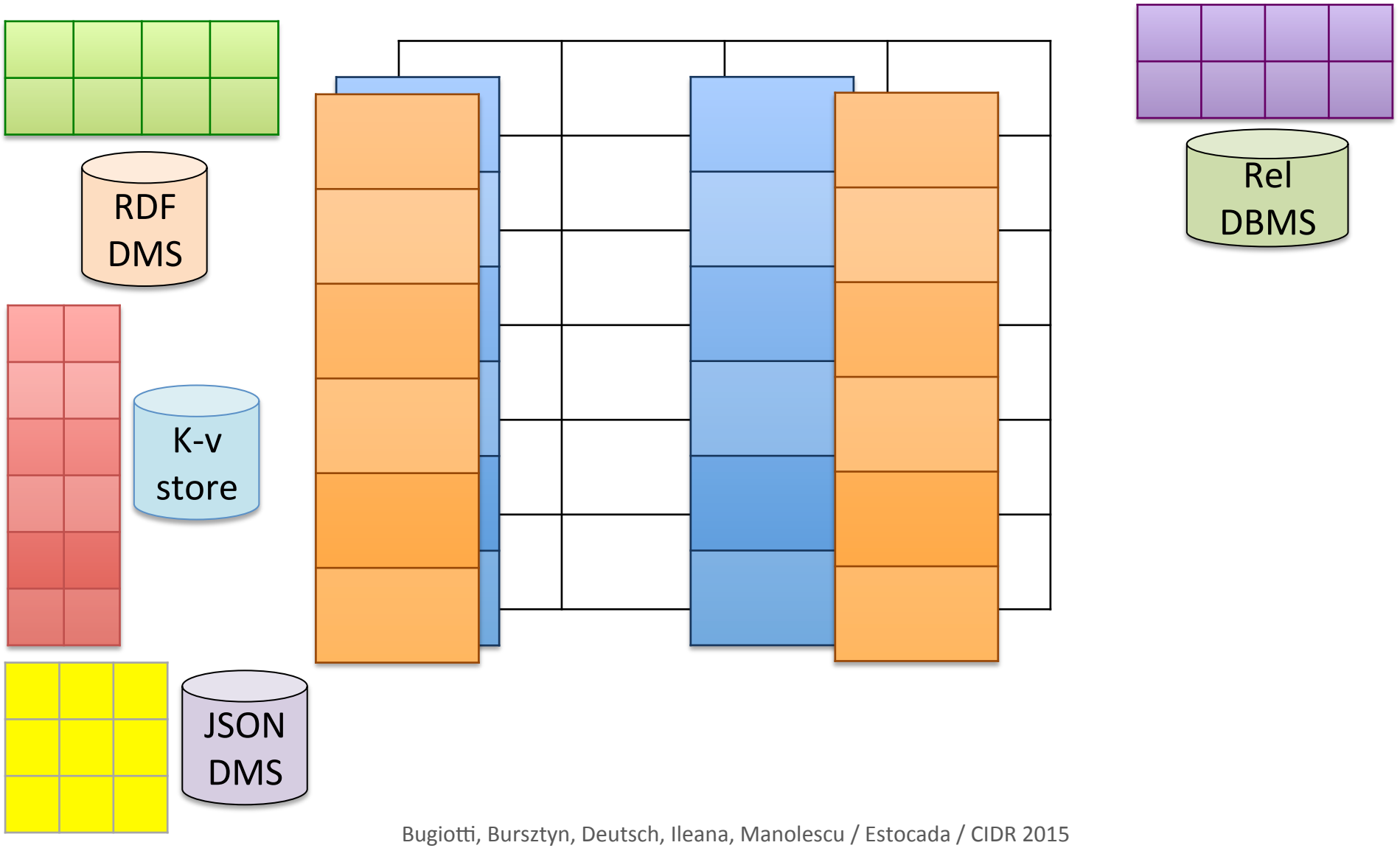
Fragments distribution across stores



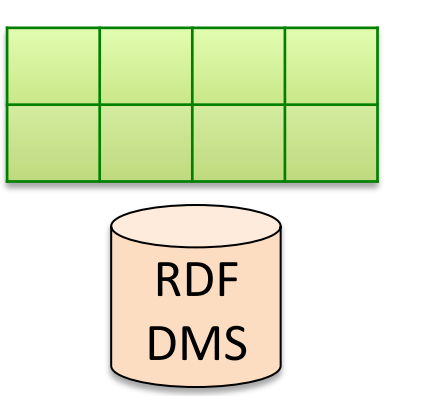
Fragments distribution across stores

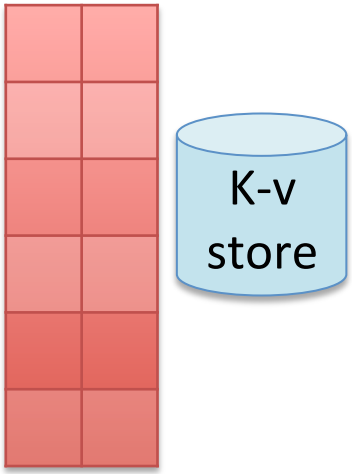


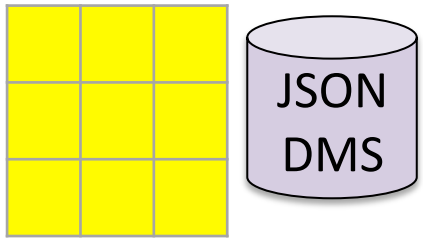
Fragments distribution across stores



Fragments distribution across stores

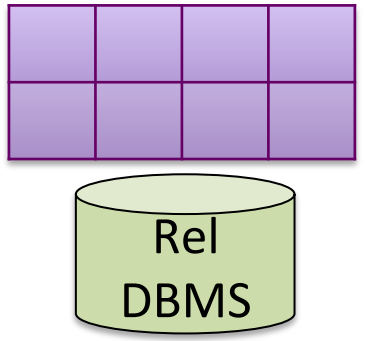


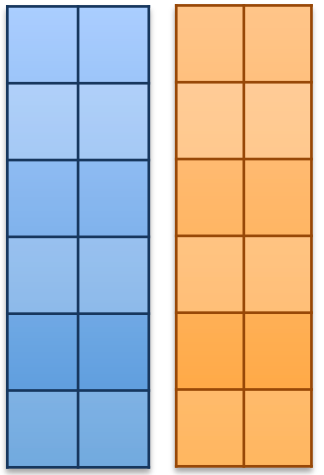




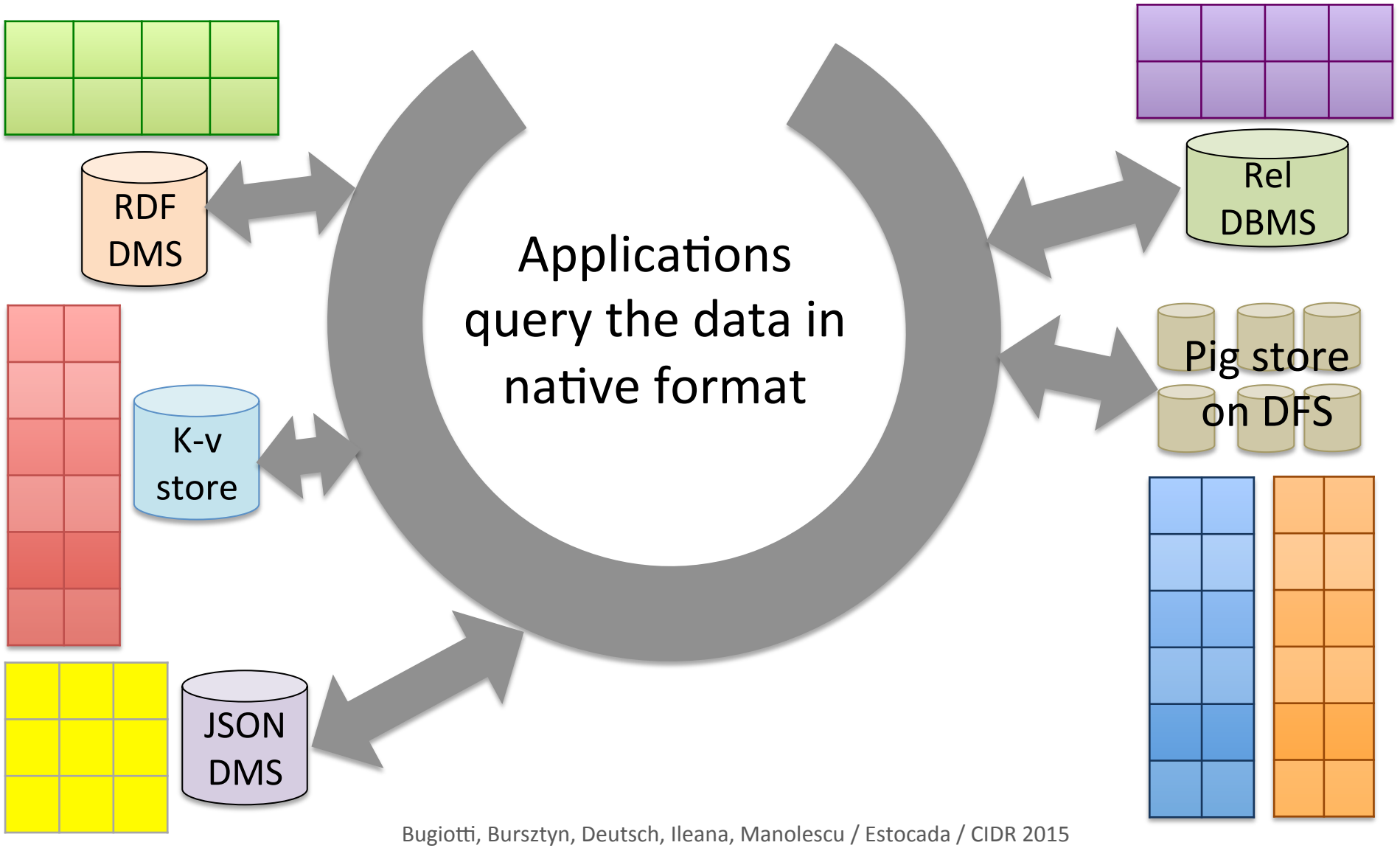
Data model translation
applied at loading

- « Dumb » translation
- The extraction logic is in the view

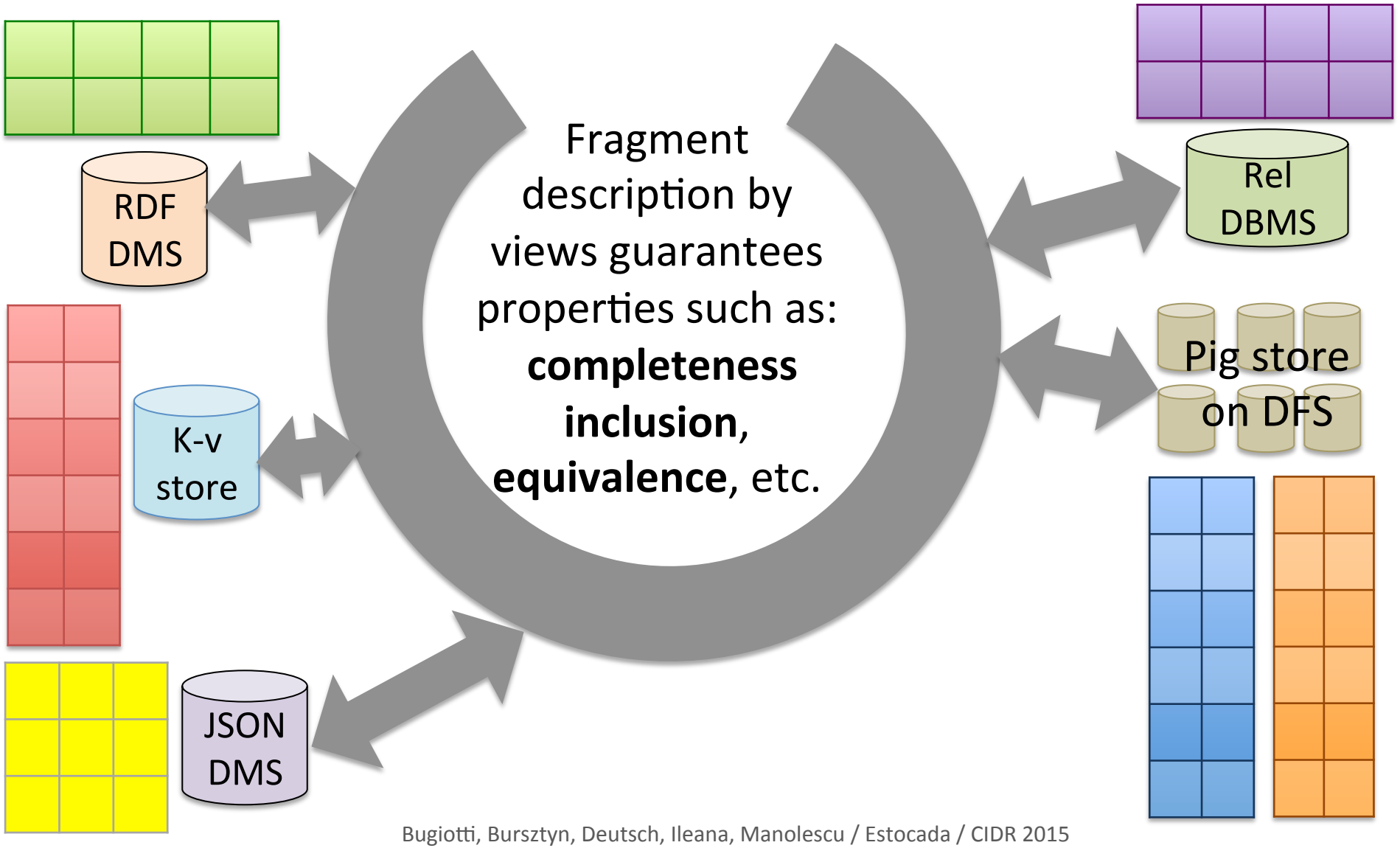




Fragments distribution across stores

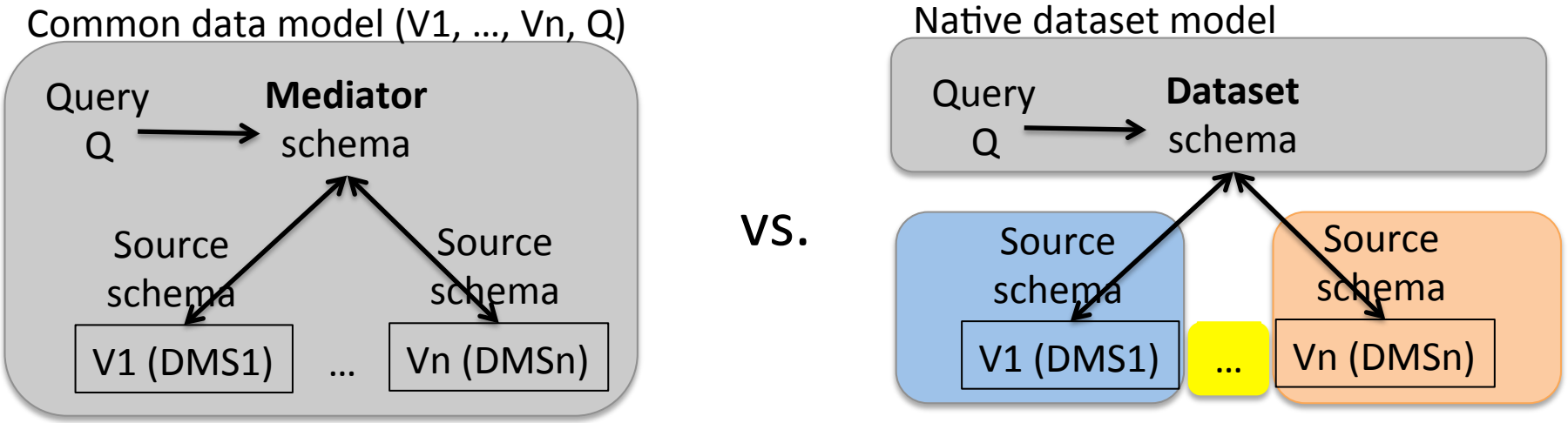


Fragments distribution across stores



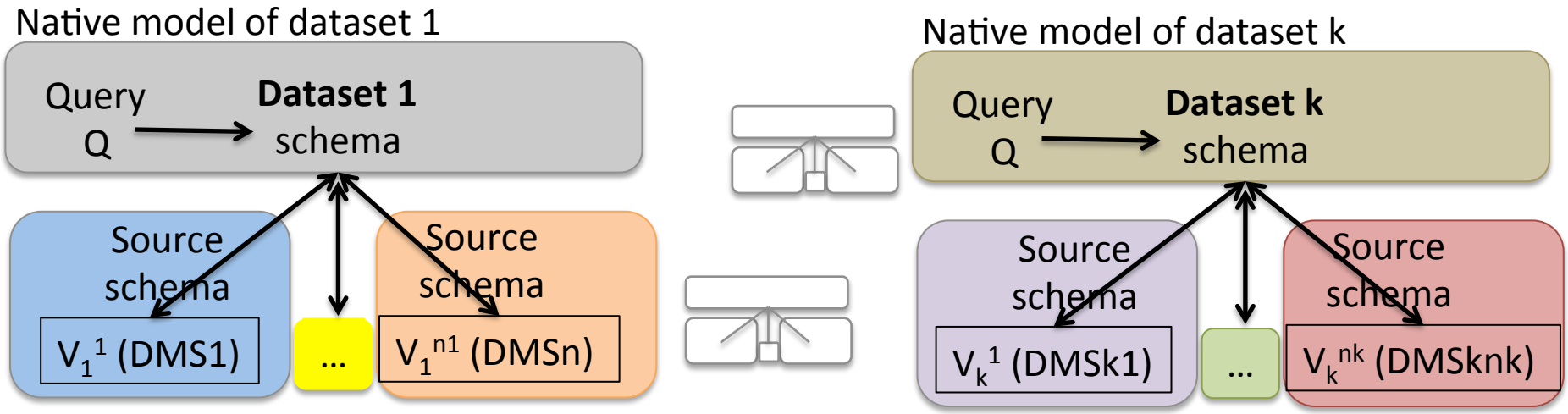
Query answering = **V**iew-**B**ased **R**ewriting

- VBR known for dramatic performance improvements
  - Basically no limit (e.g. view = query)!
  - P. Larson (2011 SIGMOD ToT): « the problem is explaining to a disappointed user that a modified view doesn't match any more! »
- Comparison with « Local As Views » mediation
  - ≠ data models



Query answering = view-based rewriting

- VBR known for dramatic performance improvements
- Comparison with « Local As Views » mediation:
  - ≠ data models
  - Side-by-side data models at the top



→ Common benefit with LAV: Applications **unaware of the fragmentation**!

→ Novel benefit: **fragments can migrate to ≠ systems and data models**

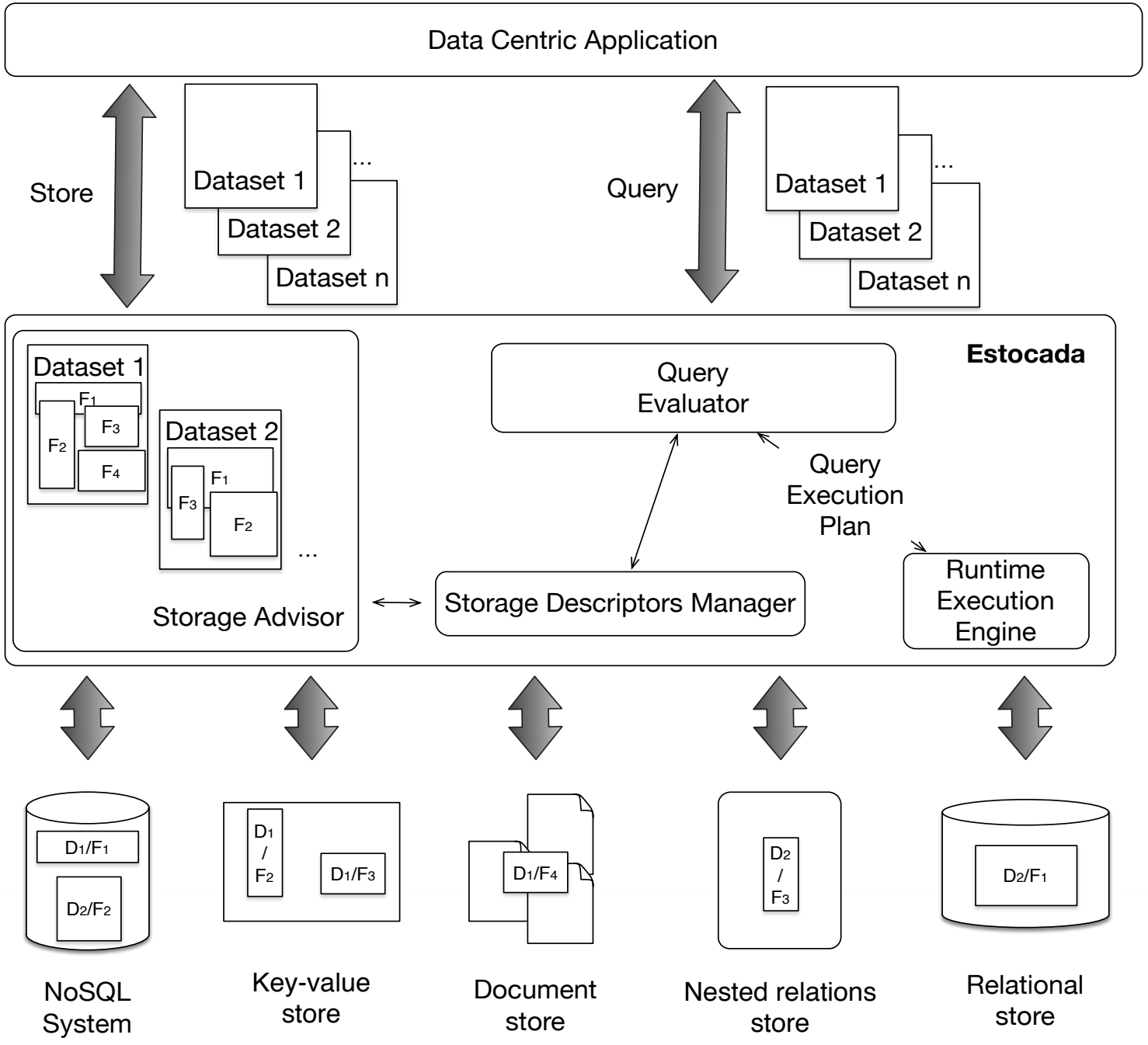
View-based rewriting with heterogeneous data models

- **Model all problems as the most complex** and solve it there
  - No! (complexity; also: The Next Data Model that will Save the World... ☺)
- Have a set of **side-by-side rewriting algorithms** (dataset model M0, DMS models M1, M2, …)
  - Impractical due to many data model combinations
- Our solution
  - Identify a **small set of « core » data models** which include the others
  - Have **side-by-side rewriting algorithms for this small core** (hint: many reuse opportunities)

VBR with heterogeneous data models

- At the core:
  - The capacity to **describe dataset properties**
    - **Structure**: (nested) tuples, trees, graphs, bags
    - **Constraints**: keys, foreign keys, inclusion dependencies (e.g. also RDF semantics)
  - A **rewriting algorithm capable of leveraging all the information** to find all equivalent query rewritings
    - Constraints enable rewritings in cases when there would be none without them!
    - Starting point: efficient Chase-and-Backchase algorithm [Ileana, Cautis, Deutsch, Katsis, SIGMOD 2014]

Estocada architecture



Estocada core modules

- VBR
  - Outputs: queries to DMSs (in their native language) + remaining integration operations
  - DMS capability descriptions exploited here.
- Runtime
  - To perform integration operations
  - For this, a single runtime (for the most expressive model, e.g. nested relations), should do
  - We may borrow one of the DMSs's runtime

What about performance?

- Select the rewriting likely to lead to the best query evaluation performance
  - **Cross-system cost model**
    - Not as crazy as it looks (cost model calibration '86; it works!)
    - Modest extension for binding patterns
- View recommendation
  - « Cross-model, cross-system data storage advisor »
  - Great progress in recent years on single-model storage (view, index etc.) recommendation
  - **Combinatorial problem** (select a subset of the possible views minimizing cost estimation)

Closest related work

- **Mixed-model VBR**
  - Agora [Manolescu, Florescu, Kossmann, VLDB 2001], Mars [Deutsch & Tannen, VLDB 2003]: XML
- Running a DMS on top of heterogeneous stores
  - **Federated databases** (Tomasic, Valduriez et al., 1996)
  - **Data integration** (wrappers / mediators) up to VIDA [Karpathiotakis, Alagiannis, Heinis, Branco, Ailamaki, CIDR 2015]
  - **Recent hybrid sy**stems [LeFevre, Sankaranarayanan, Hacigumus, Tatemura, Polyzotis, Carey, SIGMOD 2014]; [Jindal, Quiané-Ruiz, Dittrich, CIDR 2013]

Related issues

- Orthogonal concerns
  - Offering users a single integrated view
    - A la « global-as-views » data integration
  - Cleaning and extracting the data
- Not so orthogonal concerns
  - Providing common transactional properties across different DMS
  - « Failsafe mode »: use Estocada only for performance on top of a « vanilla » store
  - « Gradual »: failsafe → monitor application → recommend fragment deployment)

Sample application: smart city data integration

- Datalyse French R&D project
  - **Relational** transport database
  - **RDF** Open Data produced by the city administration (cultural artefacts, events…)
  - **Graph** social network data harvested from various applications
  - May be used with or without **log** data from city Web site and various apps
- Mid-size IT companies clueless on what to use
- Easy to be wrong by orders of magnitude

Wrap-up

- **Goal**:
  - Handle data efficiently on top of a given set of systems, even if it changes
  - Make the most out of each system
- **Method**:
  - View-based rewriting for multiple data models, under constraints
- **Status**: VBR ongoing, recommendation next

Thank you / Questions?