# The Universal Data Cube

## Progress in Konstanz

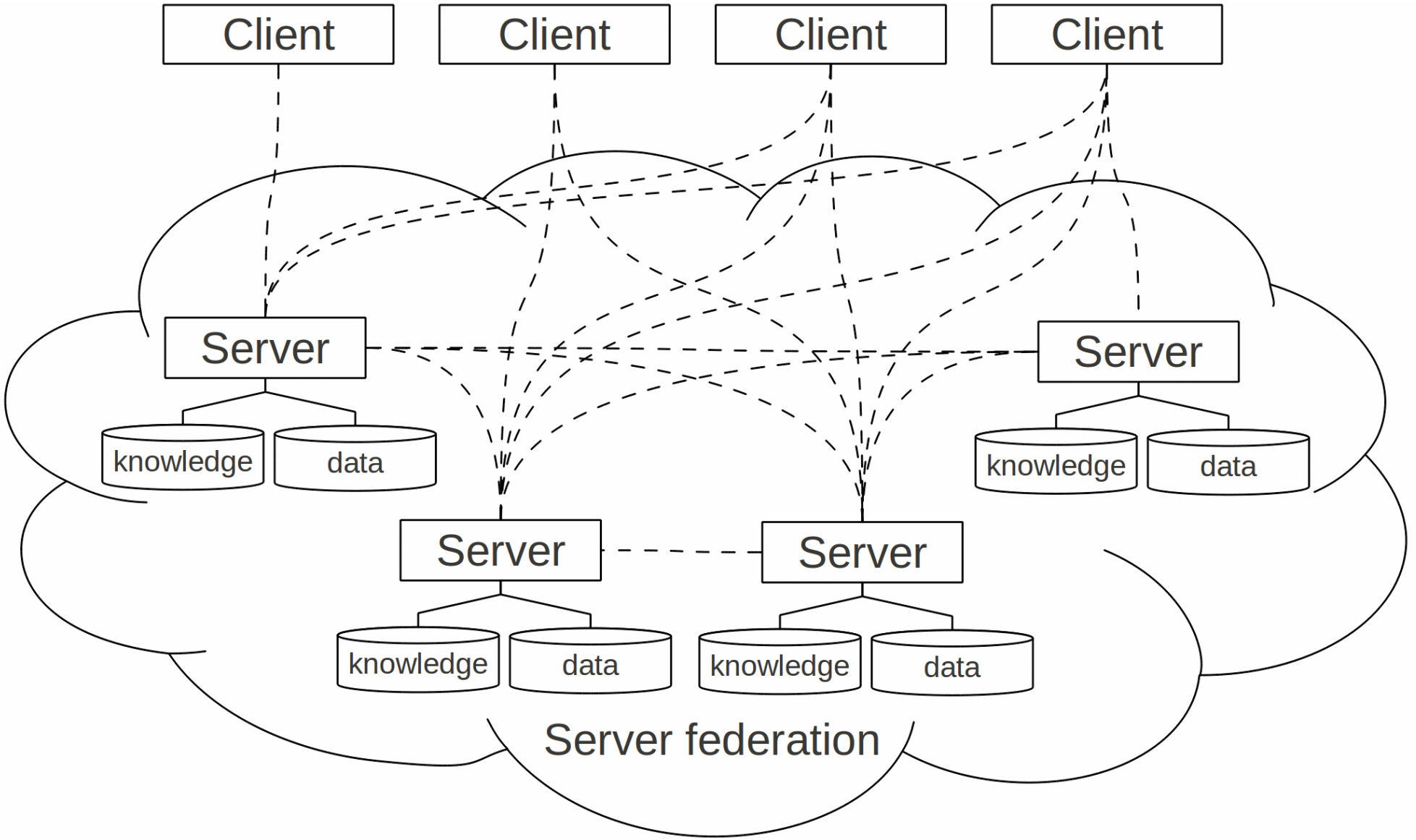
Curran Kelleher
July 23, 2010

# Outline

- The Universal Data Cube (UDC)
  - Context, Theoretical Model, Implementation
  - Example, Target Data Sources
- Collaboration Directions
- Future Plans
  - Visualization system stack design

# The Universal Data Cube

## context



# UDC Theoretical Model

Knowledge

Data

Operations

# Knowledge

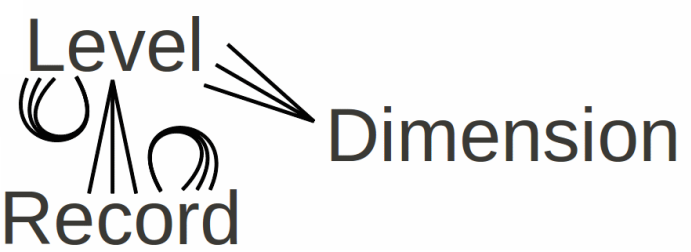
Record

# Knowledge

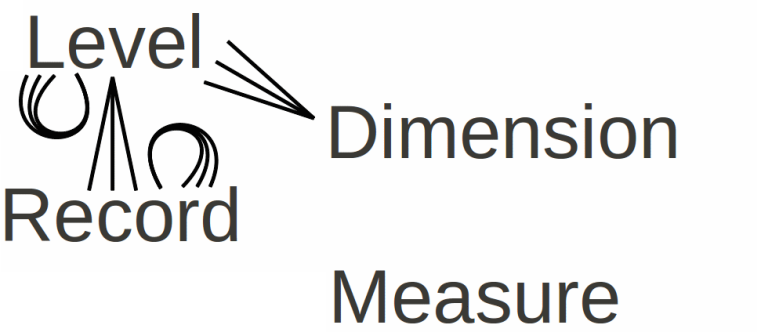
# Knowledge

# Knowledge

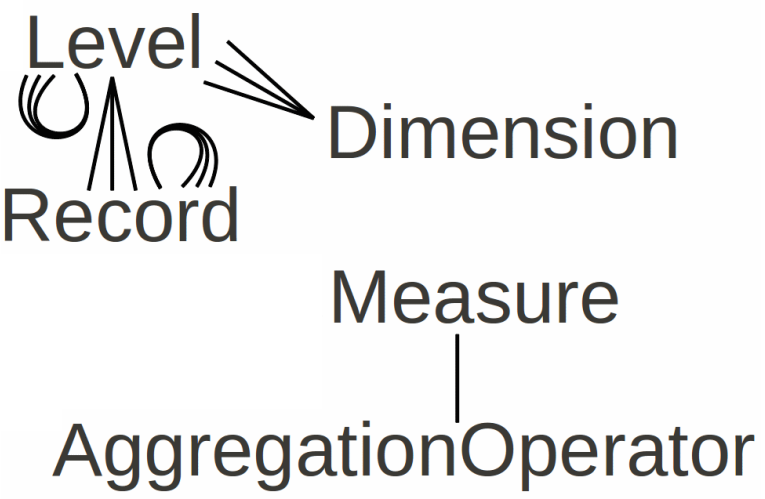
# Knowledge



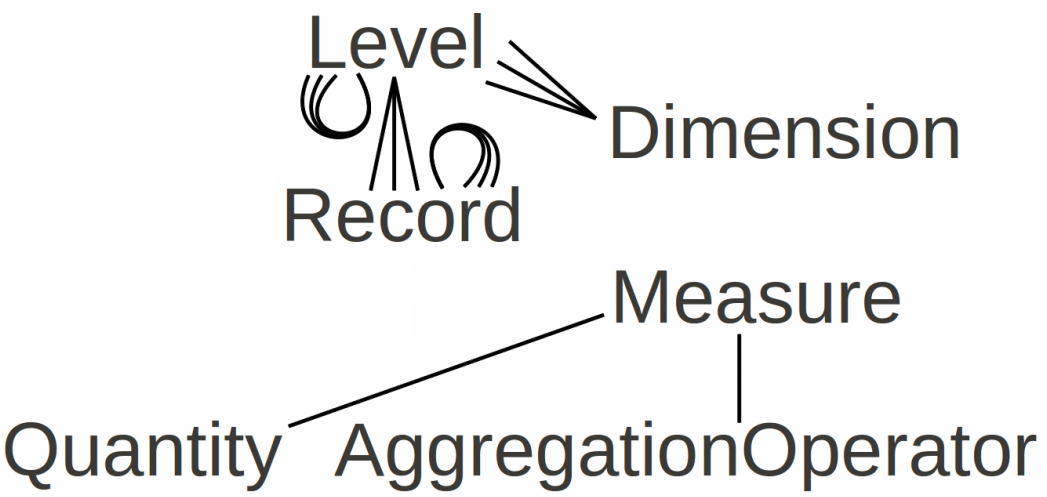
# Knowledge



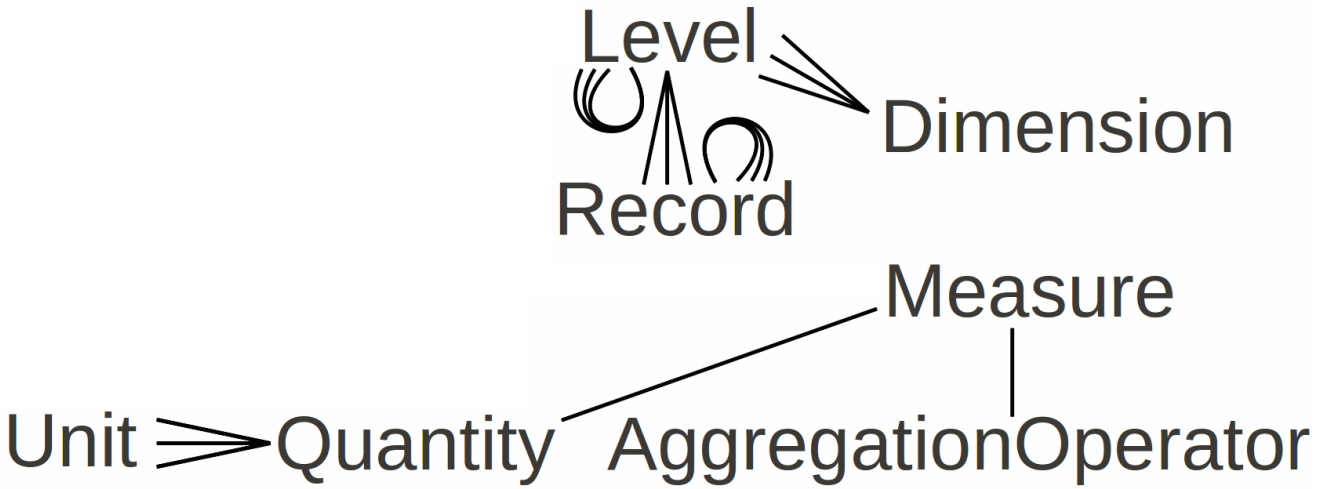
# Knowledge



# Knowledge

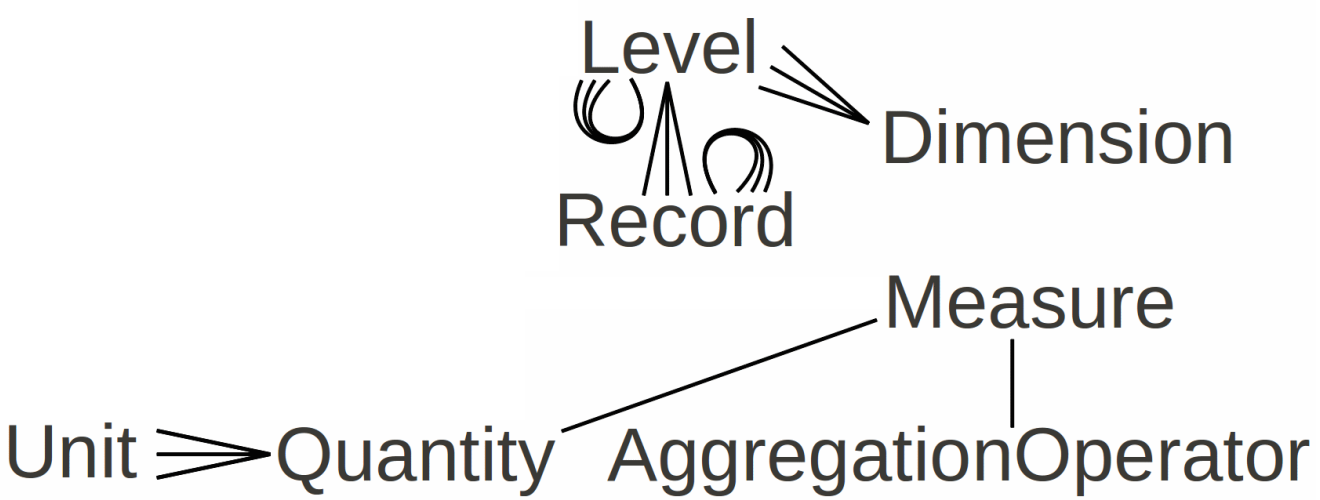

Level
Dimension
Record
Measure
Quantity
Aggregation
Operator

# Knowledge

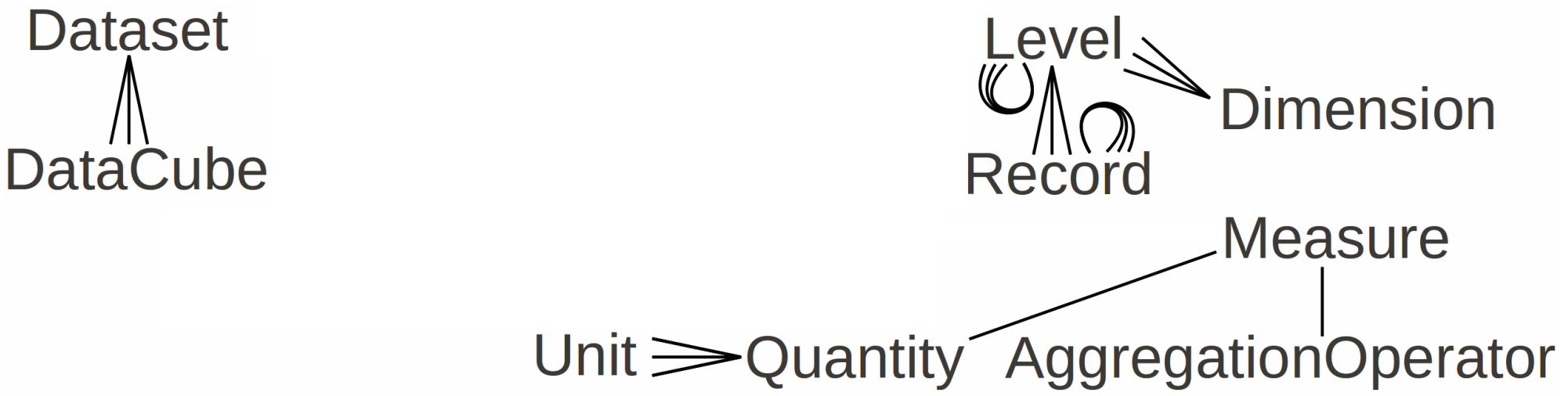

Level
Dimension
Record
Measure
Unit
Quantity
Aggregation
Operator

# Knowledge

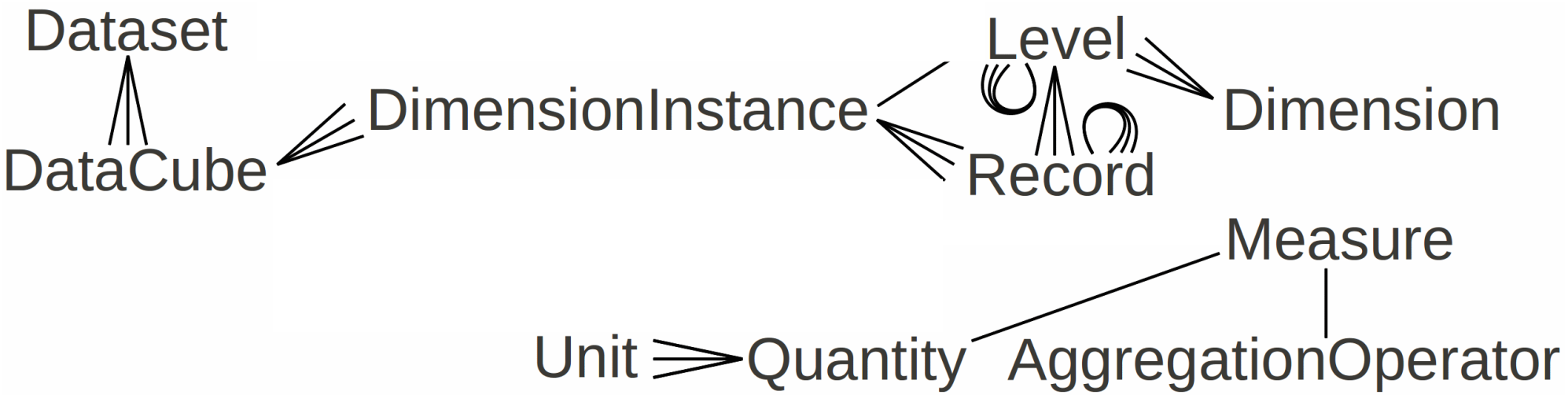
Dataset



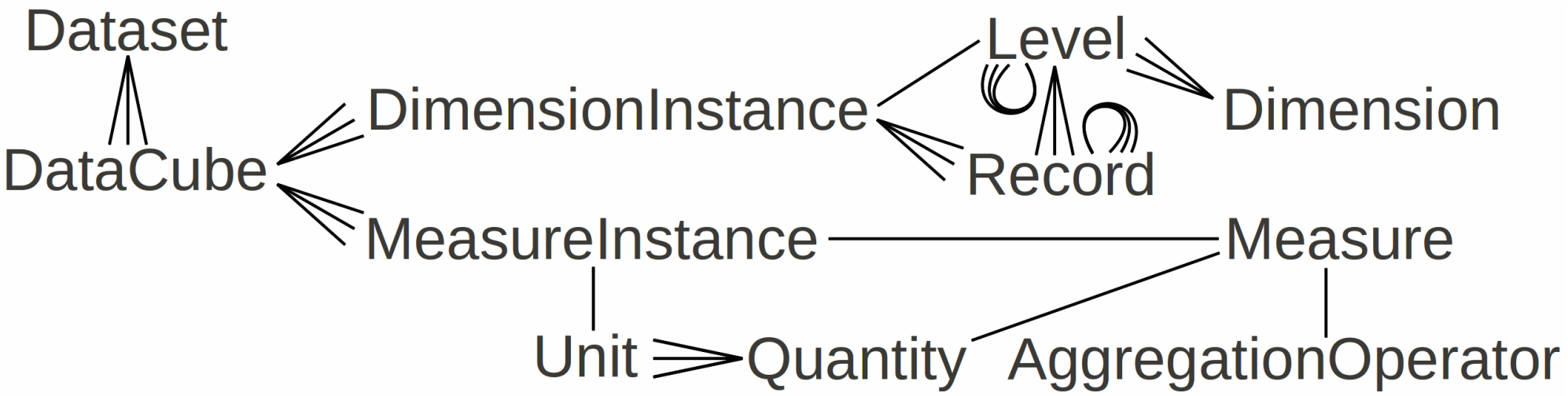
# Knowledge


Dataset
DataCube
Level
Dimension
Record
Measure
Unit
Quantity
AggregationOperator

# Knowledge

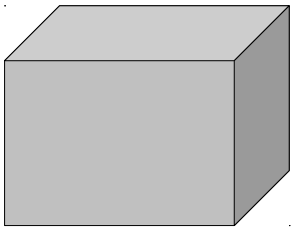

Dataset
DataCube
DimensionInstance
Level
Record
Dimension
Measure
Unit
Quantity
AggregationOperator

# Knowledge

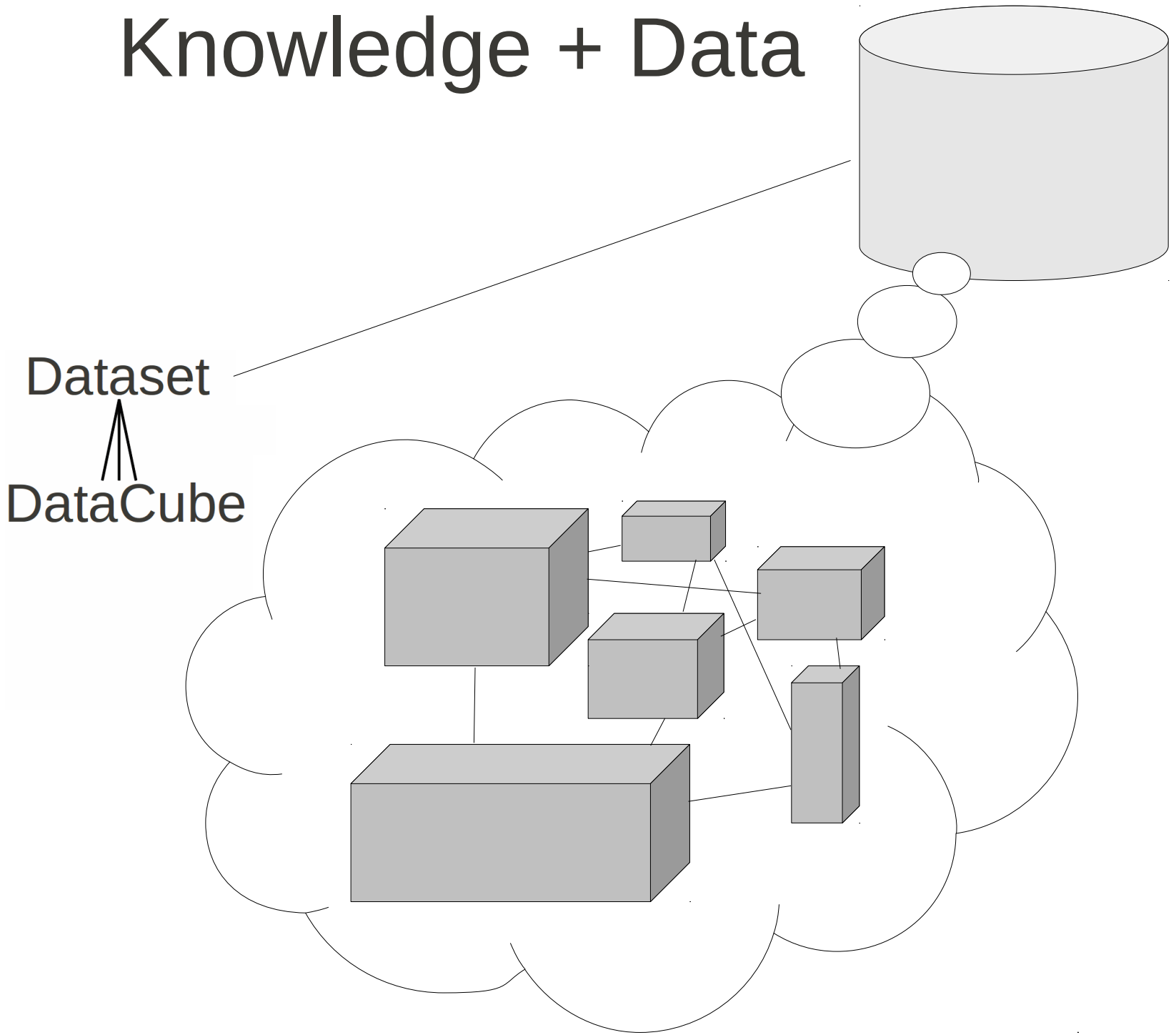

Dataset
DataCube
DimensionInstance
MeasureInstance
Level
Record
Dimension
Measure
Unit
Quantity
AggregationOperator

# Knowledge + Data

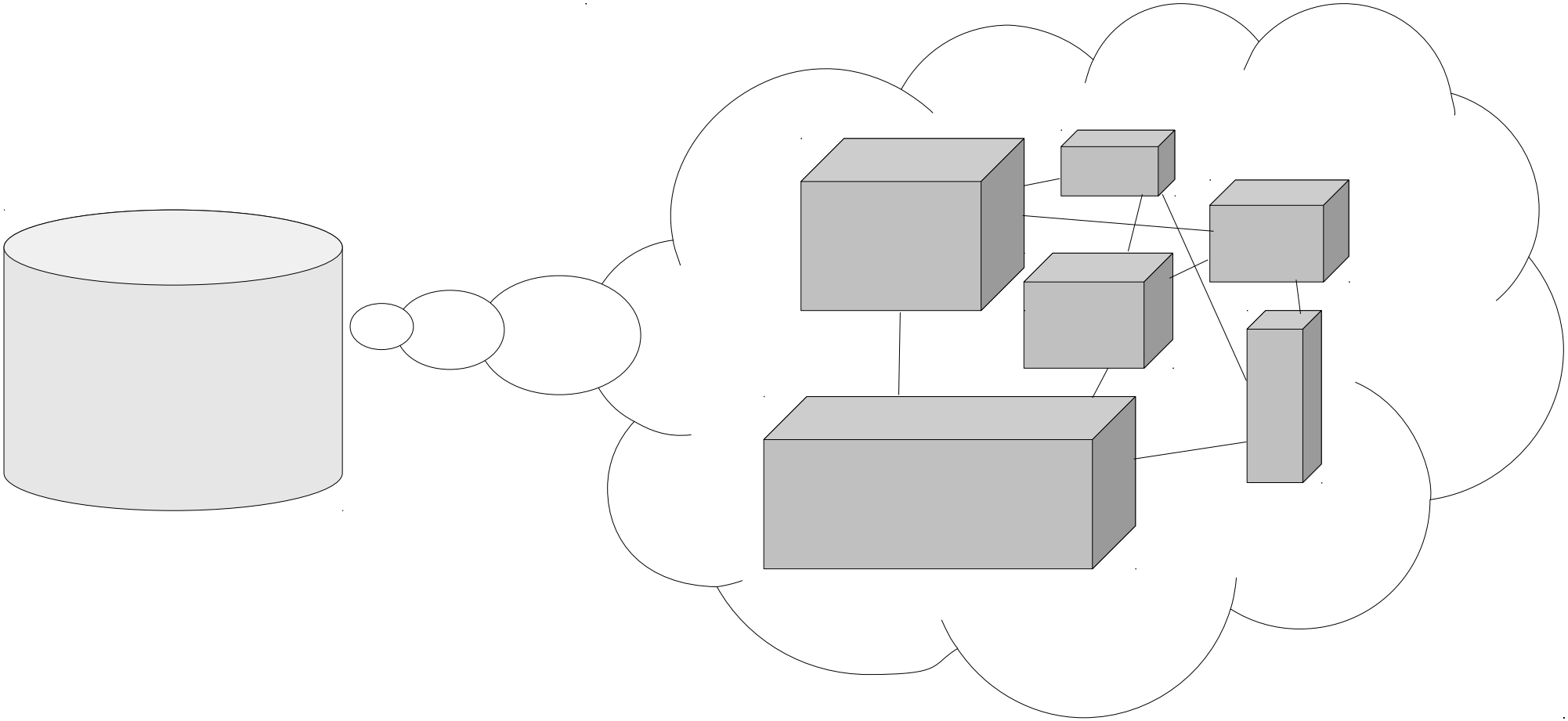
DataCube



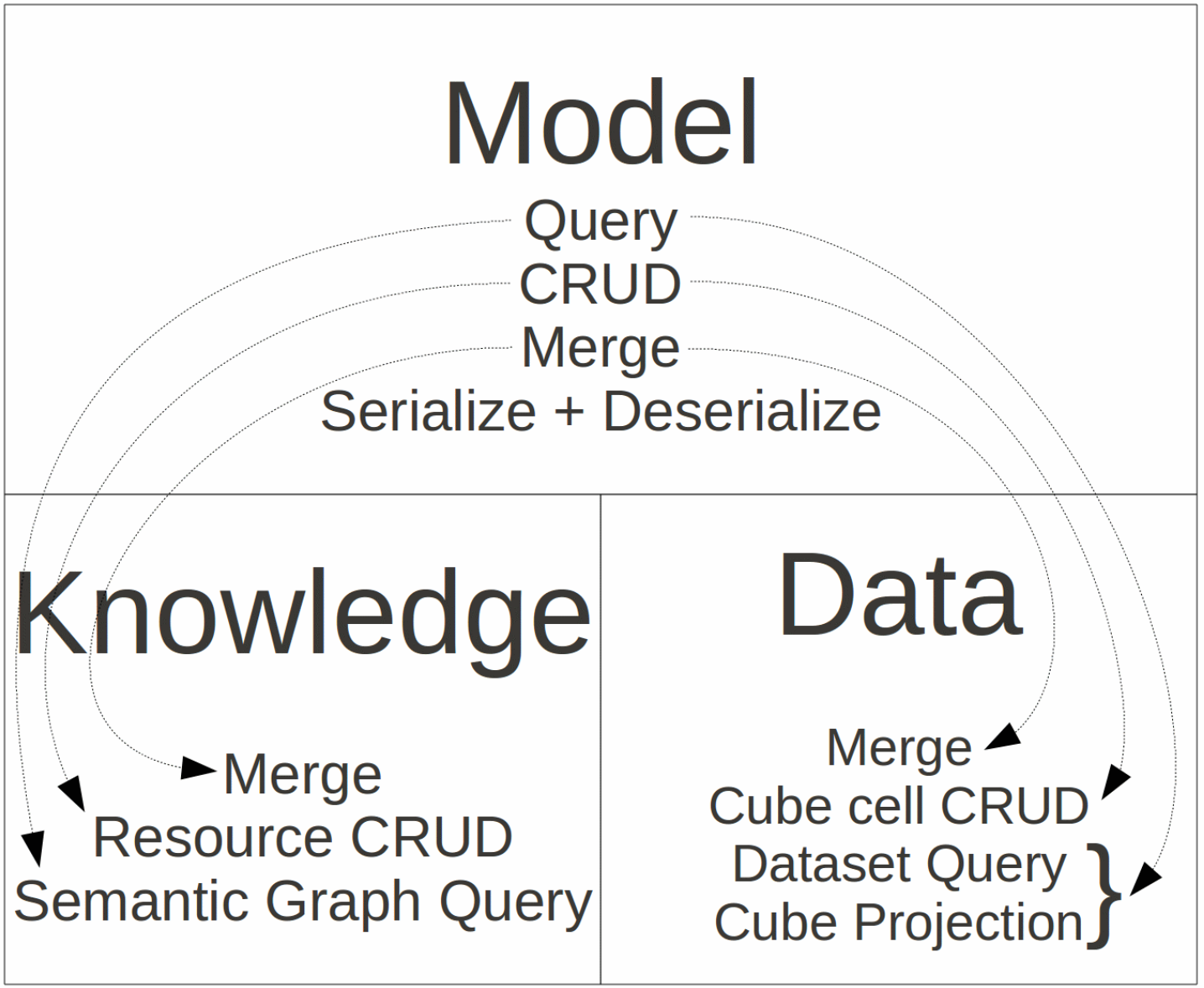
# Knowledge + Data



# Data



# Operations


Model
Query
CRUD
Merge
Serialize + Deserialize
Knowledge
Merge
Resource CRUD
Semantic Graph Query
Data
Merge
Cube cell CRUD
Dataset Query
Cube Projection

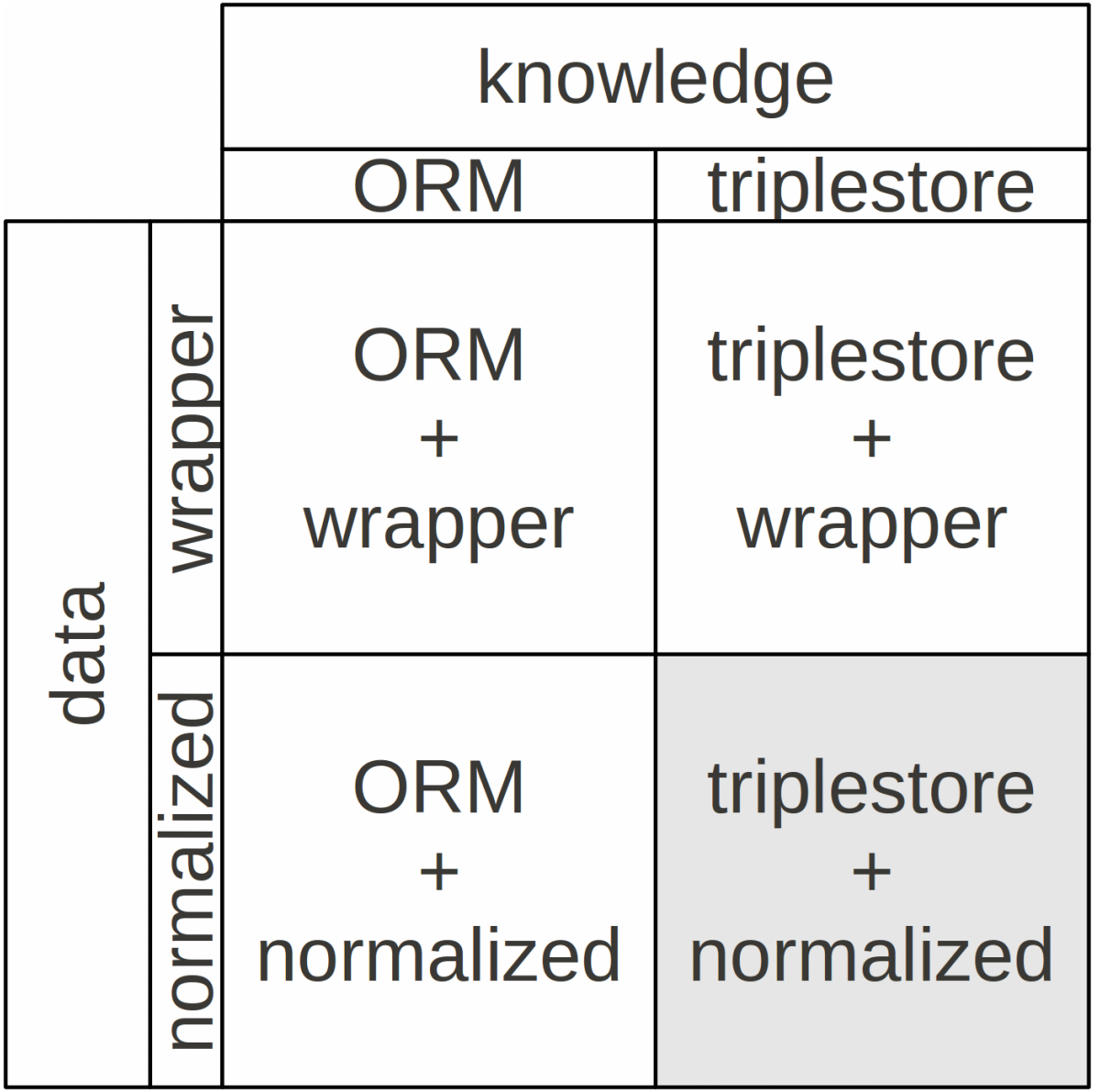
# UDC Implementation

Knowledge

Data

Operations

# Knowledge + Data Persistence Implementation

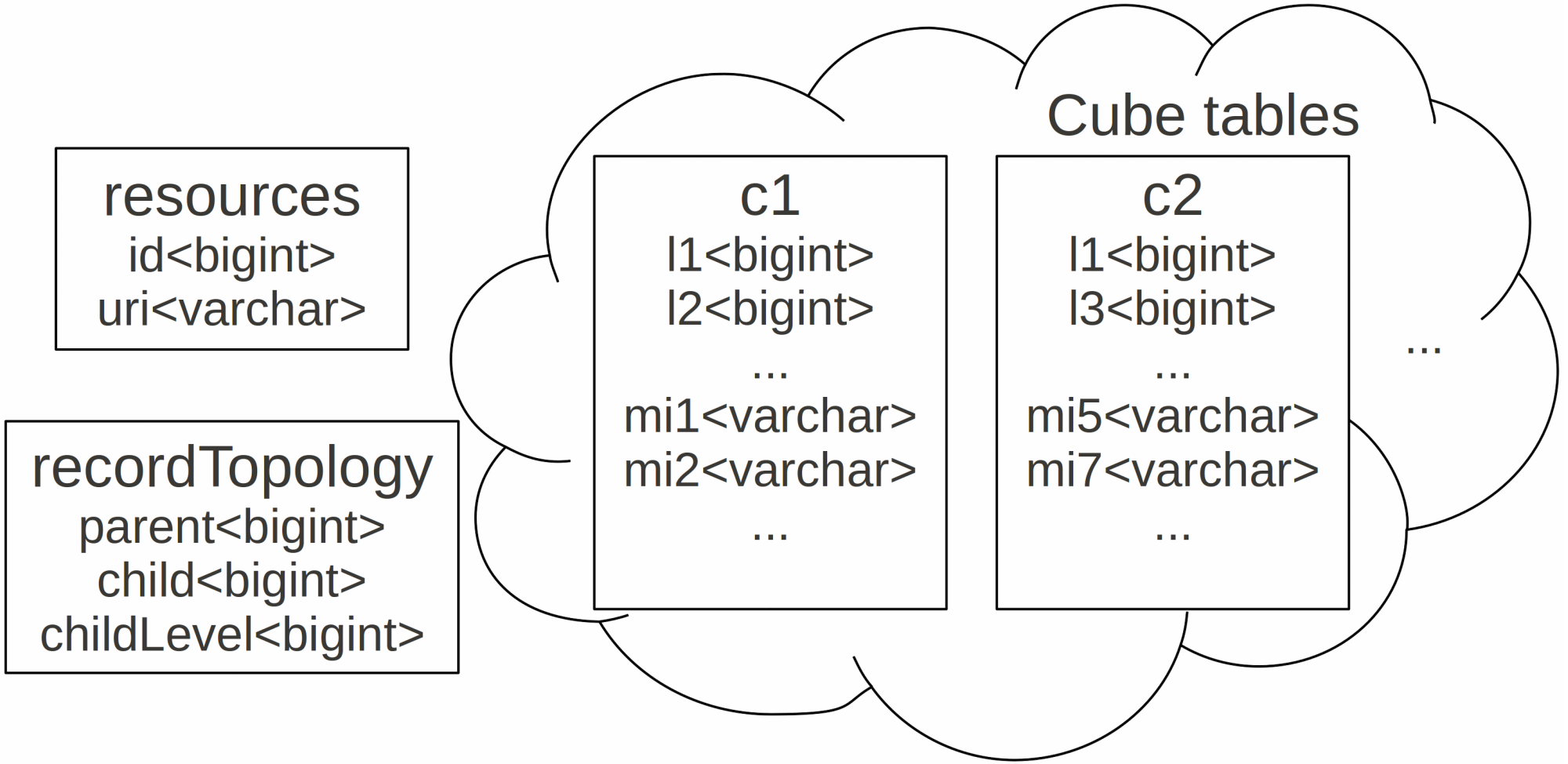


# Operations Implementation

<table>
<tr><th></th><th>Knowledge</th><th>Data</th></tr>
<tr><td>Create</td><td rowspan="4">JENA</td><td rowspan="5">udc-sql</td></tr>
<tr><td>Read</td></tr>
<tr><td>Update</td></tr>
<tr><td>Delete</td></tr>
<tr><td>Query</td><td>SPARQL</td></tr>
</table>

# udc-sql

normalized schema



# An Example UDC Model

visualized as code

```
private UDCModel createUDCModel() {
    return new UDCModel(createUDCKnowledge(), createUDCData());
}
private Knowledge createUDCKnowledge() {
    //an in-memory knowledge implementation
    return new IMKnowledge();
}
private Data createUDCData() {
    //a normalized sql data implementation
    // inside an in-memory HQSLDB database
    jdbcDataSource ds = new jdbcDataSource();
    ds.setDatabase("jdbc:hsqldb:mem:sqltest");
    ds.setUser("sa"); ds.setPassword("");
    return new UDCDataSQL(ds);
}
```

```
public UDCModel generateTestModel() {
UDCModel model = createUDCModel();
String domain = "http://datacubes.org";
String dimensionsURI = domain + "/dimensions";
String quantitiesURI = domain + "/quantities";
String unitsURI = domain + "/units";
String measuresURI = domain + "/measures";
String aggregationOperatorsURI = domain + "/aggregationOperators";
String datasetsURI = domain + "/datasets";
String dataCubesURI = domain + "/dataCubes";

String timeURI = dimensionsURI + "/time";
Dimension time = model.createDimension(timeURI);
time.setName(intlName("Time", "en", "Zeit", "de"));

String spaceURI = dimensionsURI + "/space";
Dimension space = model.createDimension(spaceURI);
space.setName(intlName("Space", "en", "Raum", "de"));

String yearsURI = timeURI + "/years";
Level years = time.createLevel(yearsURI);
years.setName(intlName("Year", "en", "Jahr", "de"));
years.setNamePlural(intlName("Years", "en", "Jahre", "de"));

String monthsURI = timeURI + "/months";
Level months = time.createLevel(monthsURI);
months.setName(intlName("Month", "en", "Monat", "de"));
months.setNamePlural(intlName("Months", "en", "Monate", "de"));
months.addParent(years);

String countriesURI = spaceURI + "/countries";
Level countries = space.createLevel(countriesURI);
countries.setName(intlName("Country", "en", "Land", "de"));
countries.setNamePlural(intlName("Countries", "en", "Lände", "de"));

String usStatesURI = spaceURI + "/usStates";
Level usStates = space.createLevel(usStatesURI);
usStates.setName(intlName("State", "en", "Staat", "de"));
usStates.setNamePlural(intlName("States", "en", "Staaten", "de"));
usStates.addParent(countries);

String usCountiesURI = spaceURI + "/usCounties";
Level usCounties = space.createLevel(usCountiesURI);
usCounties.setName(intlName("County", "en"));
usCounties.setNamePlural(intlName("County", "en"));
usCounties.addParent(usStates);

String bundesländeURI = spaceURI + "/bundesländе";
Level bundesländе = space.createLevel(bundesländeURI);
bundesländе.setName(intlName("Bundesland", "de", "State", "en"));
bundesländе.setNamePlural(intlName("Bundesländе", "de", "States", "en"));
bundesländе.addParent(countries);

String regierungsbezirkeURI = spaceURI + "/regierungsbezirke";
Level regierungsbezirke = space.createLevel(regierungsbezirkeURI);
regierungsbezirke.setName(intlName(
        "Regierungsbezirk", "de","Administrative Region", "en"));
regierungsbezirke.setNamePlural(intlName(
        "Regierungsbezirke", "de","Administrative Regions", "en"));
regierungsbezirke.addParent(bundesländе);

String landkreiseURI = spaceURI + "/landkreise";
Level landkreise = space.createLevel(landkreiseURI);
landkreise.setName(intlName("Landkreis", "de", "District", "en"));
landkreise.setNamePlural(intlName("Landkreiss", "de", "Districts", "en"));
landkreise.addParent(regierungsbezirke);

String year1991URI = yearsURI + "/1991";
Record year1991 = years.createRecord(year1991URI);
year1991.setName(intlName("1991", "en"));

String year1990URI = yearsURI + "/1990";
Record year1990 = years.createRecord(year1990URI);
year1990.setName(intlName("1990", "en"));
year1990.setNext(year1991);

String newYorkURI = usStatesURI + "/NewYork";
Record newYork = usStates.createRecord(newYorkURI);
newYork.setName(intlName("New York", "en"));
newYork.addParent(usa);

String massachusettsURI = usStatesURI + "/Massachusetts";
Record massachusetts = usStates.createRecord(massachusettsURI);
massachusetts.setName(intlName("Massachusetts", "en"));
massachusetts.addParent(usa);

String middlesexURI = usCountiesURI + "/Middlesex";
Record middlesex = usCounties.createRecord(middlesexURI);
middlesex.setName(intlName("Middlesex", "en"));
massachusetts.addParent(massachusetts);

String currencyURI = quantitiesURI + "/Currency";
Quantity currency = model.createQuantity(currencyURI);
currency.setName(intlName("Currency", "en"));

String numberOfPeopleURI = quantitiesURI + "/NumberOfPeople";
Quantity numberOfPeople = model.createQuantity(numberOfPeopleURI);
numberOfPeople.setName(intlName("Number of People", "en"));

String usDollarsURI = unitsURI + "/usDollars";
Unit usDollars = model.createUnit(usDollarsURI);
usDollars.setName(intlName("U. S. Dollars", "en"));
usDollars.setQuantity(currency);

String personsURI = unitsURI + "/persons";
Unit persons = model.createUnit(personsURI);
persons.setName(intlName("Persons", "en"));
persons.setQuantity(currency);

String deutschlandURI = countriesURI + "/Deutschland";
Record deutschland = countries.createRecord(deutschlandURI);
deutschland.setName(intlName("Deutschland", "de", "Germany", "en"));
usa.addAlternateName(locName("Federal Republic of Germany", "en"));
usa.addAlternateName(locName("Bundesrepublik Deutschland", "de"));

String usaURI = countriesURI + "/USA";
Record usa = countries.createRecord(usaURI);
usa.setName(intlName("United States of America", "en",
        "Vereinigte Staaten von Amerika", "de"));
usa.addAlternateName(locName("USA", "en"));
usa.addAlternateName(locName("United States", "en"));
usa.addAlternateName(locName("The States", "en"));
usa.addAlternateName(locName("The U. S.", "en"));
usa.addAlternateName(locName("Amerika", "de"));
usa.addAlternateName(locName("Die Vereinigten Staaten", "de"));

String badenWürttembergURI = bundesländeURI + "/Baden-Württemberg";
Record badenWürttemberg = bundesländе.createRecord(badenWürttembergU
badenWürttemberg.setName(intlName("Baden-Württemberg", "de"));
badenWürttemberg.addParent(deutschland);

String freiburgURI = regierungsbezirkeURI + "/Freiburg";
Record freiburg = regierungsbezirke.createRecord(freiburgURI);
freiburg.setName(intlName("Freiburg", "de"));
freiburg.addParent(badenWürttemberg);

String konstanzURI = landkreiseURI + "/Konstanz";
Record konstanz = landkreise.createRecord(konstanzURI);
konstanz.setName(intlName("Konstanz", "de", "Constance", "en"));
konstanz.addParent(freiburg);

String sumURI = aggregationOperatorsURI + "/sum";
AggregationOperator sum = model.createAggregationOperator(sumURI);
sum.setName(intlName("Sum", "en", "Summe", "de"));

String avgURI = aggregationOperatorsURI + "/average";
AggregationOperator avg = model.createAggregationOperator(av
avg.setName(intlName("Average", "en", "Durchschnitt", "de"))

String incomeURI = measuresURI + "/income";
Measure income = model.createMeasure(incomeURI);
income.setName(intlName("Income", "en", "Einkommen", "de"));
income.setQuantity(currency);
income.setAggregationOperator(avg);

String populationURI = measuresURI + "/population";
Measure population = model.createMeasure(populationURI);
population.setName(intlName("Population", "en", "Einwohner", "de"));
population.setQuantity(numberOfPeople);
population.setAggregationOperator(sum);

String blsURI = datasetsURI + "/bls";
Dataset bls = model.createDataset(blsURI);
bls.setName(intlName("The Bureau of Labor Statistics Employment Dataset", "en"));
bls.setCreator("U.S. Bureau of Labor Statistics");
bls.setDescription("The Bureau of Labor Statistics Employment Dataset. "
    + "Downloaded and imported by the Institute for Visualization "
    + "and Perception Research at UMass Lowell."
    + " Original data files are located at "
    + "ftp://ftp.bls.gov/pub/special.requests/cew/");

blsYearsByStates.setValues(cellLocation(year1990, massachusetts),measureInstances(blsIncome, blsPopulation),values(36952,6022639));
blsYearsByStates.setValues(cellLocation(year1991, massachusetts),measureInstances(blsIncome, blsPopulation),values(37563,6023471));
blsYearsByStates.setValues(cellLocation(year1990, newYork),measureInstances(blsIncome, blsPopulation),values(37835,6084523));
blsYearsByStates.setValues(cellLocation(year1991, newYork),measureInstances(blsIncome, blsPopulation),values(38463,9057371));
}
```

# Target Data Sources

## for transformation and import

# The Bureau of Labor Statistics Employment Dataset

as dimensions and measures

- Time
  - Years, quarters, months
- Space
  - US, US States, US Counties
- Industry
  - NAICS Industry Heirarchy
- Ownership
  - Government (Federal, State, Local), Private
- Employment, Annual Pay, Total Wages, etc.

# The Eurostat Employment Dataset

as dimensions and measures

- Time
  - Years, quarters, months
- Space
  - Europe, Countries, Country Regions
- Gender
- Age Range
- Employment, Annual Pay, Total Wages, etc.

# The Bible

as dimensions and measures

- Document Region
  - Books, Chapters, Verses, Sentences, Words
- Measures
  - Readability, Sentiment, Complexity, Word count
  - Sentence length, noun / verb ratio
  - simpson's index (vocabulary richness)
  - Thematic content, word stem frequencies
  - Parse tree branching factor, passive/active

# Genomic Data

as dimensions and measures

- DNA Region
  - Species category, Genome, Chromosome, Gene, Gene cluster, Overlapping gene region, Protein-Coding Region, Codon, Base
- Genomic statistics
  - Number of base pairs per base type
  - Number of Codons per amino acid type
  - Number of proteins coded in region
  - Number of overlapping gene regions

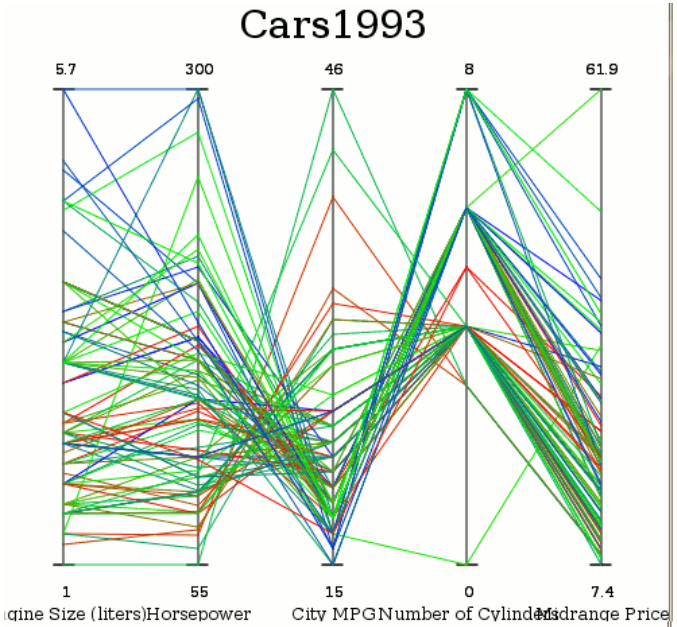
# Geospatial Movement Data

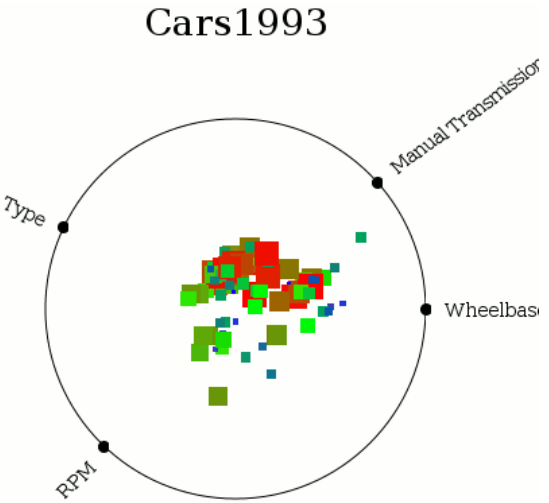
as dimensions and measures

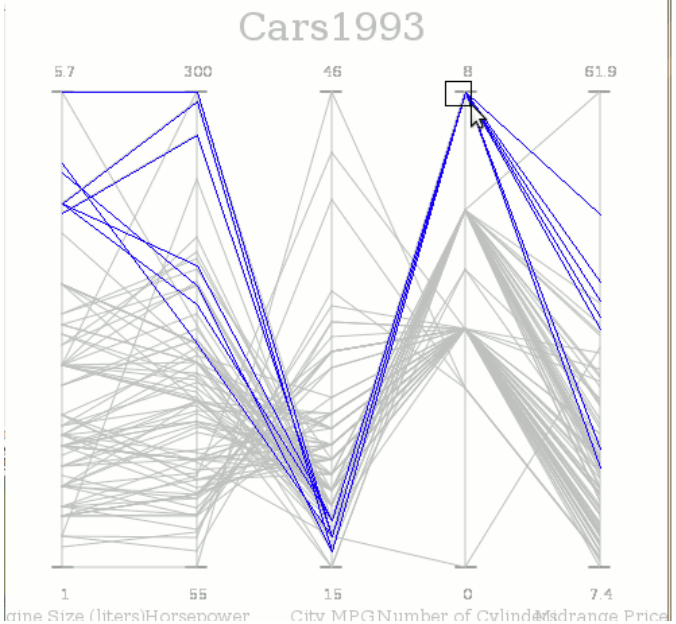
- Time
  - Years, quarters, months, days, minutes, seconds
- Space
  - Political boundary region hierarchies
  - Algorithmic region hierarchies (quadtree, hextree)
- Agent Category
  - Type of moving thing (animal, human, vehicle, etc)
- Movement Path
  - Paths, regional subpaths, location observation
- Lat, Long, Direction, Velocity, Density
  - Or other measures derivable from a movement path set

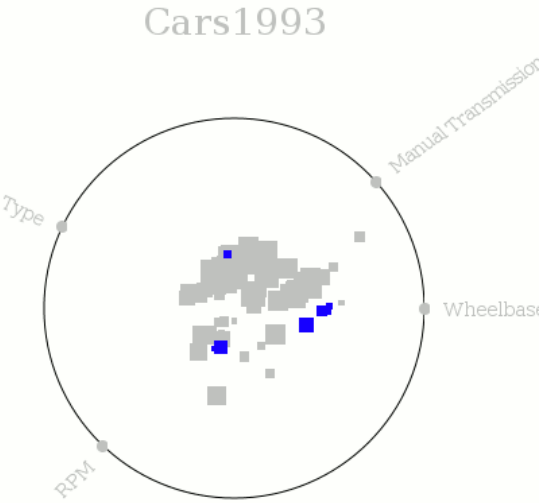
# Collaboration Directions

# JyVis


Cars1993
5.7
300
46
8
61.9
1
55
15
0
7.4
Horsepower
City MPG
Number of Cylinders

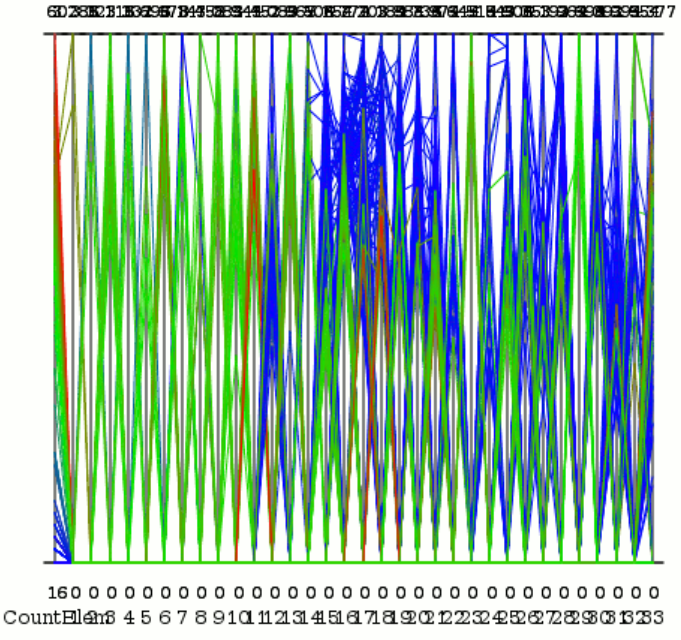

Cars1993
Manual Transmissior
Type
RPM


Cars1993
5.7
300
46
8
61.9
1
55
15
0
7.4
Horsepower
City MPG
Number of Cylinders

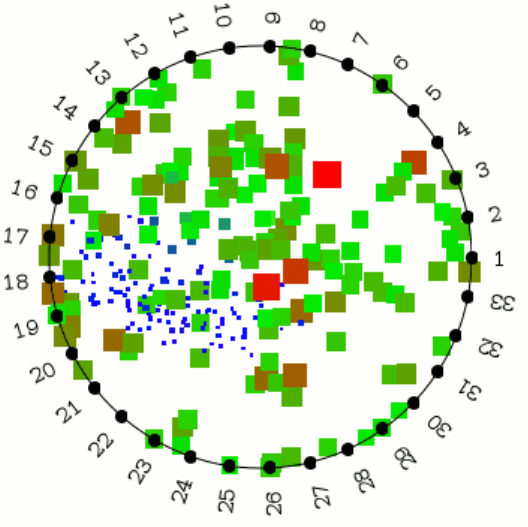

Cars1993
Manual Transmissior
Type
RPM

# JyVis + Subspace Clustering

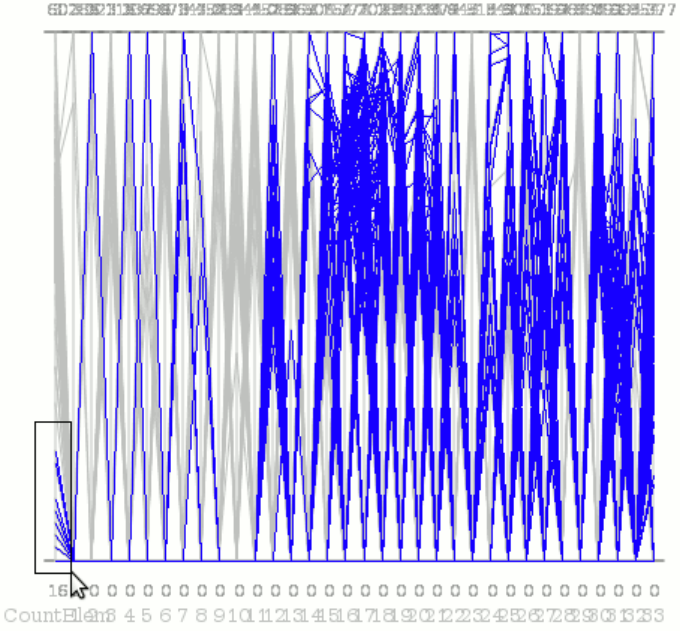
breast-SCHISM-weights_inverted



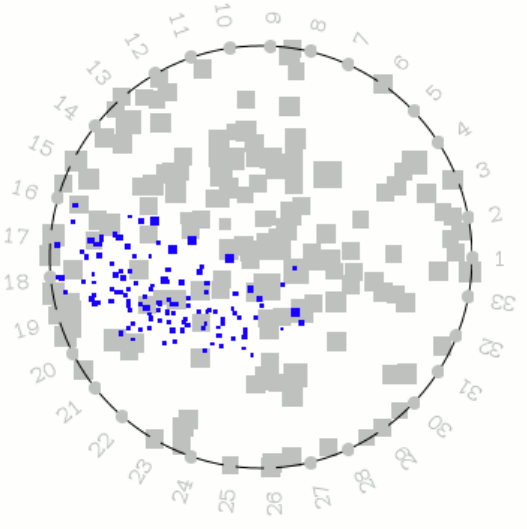
breast-SCHISM-weights_inverted



breast-SCHISM-weights_inverted



breast-SCHISM-weights_inverted

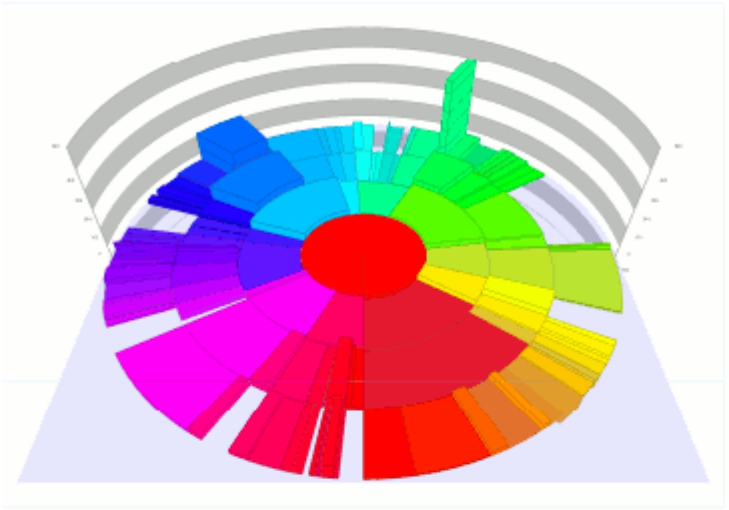


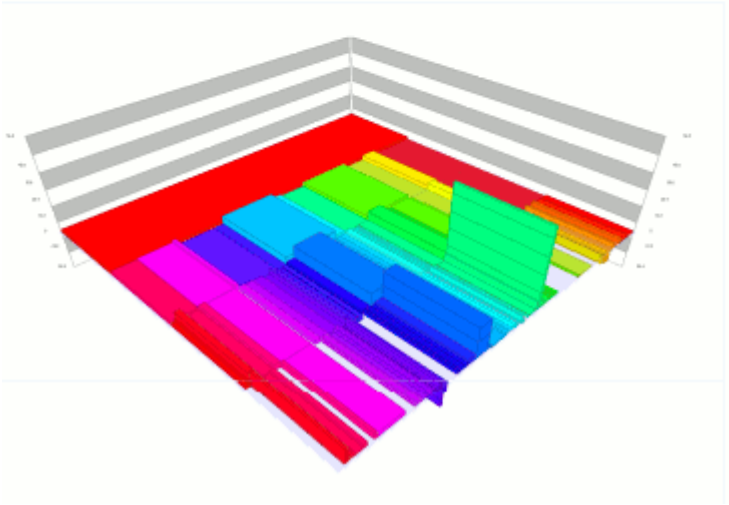
# Collaboration Idea

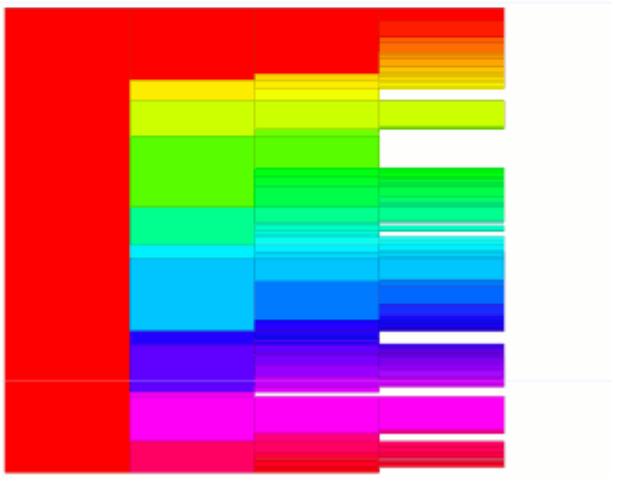
- Develop a software library providing
  - An interactive graphics canvas
  - A rich set of interactive visualization primitives
  - A generalized brushed selection model
  - Efficient implementations of various selection modes using spatial indices
- Key: it is data-model-independent
  - Could be used in a subspace cluster / hierarchical data cube / data table visualization tool

# Data Quality Visualization for Multivariate Hierarchic Data

Tatiana Tekusova, Martin Knuth, Tobias Schreck, Jorn Kohlhammer



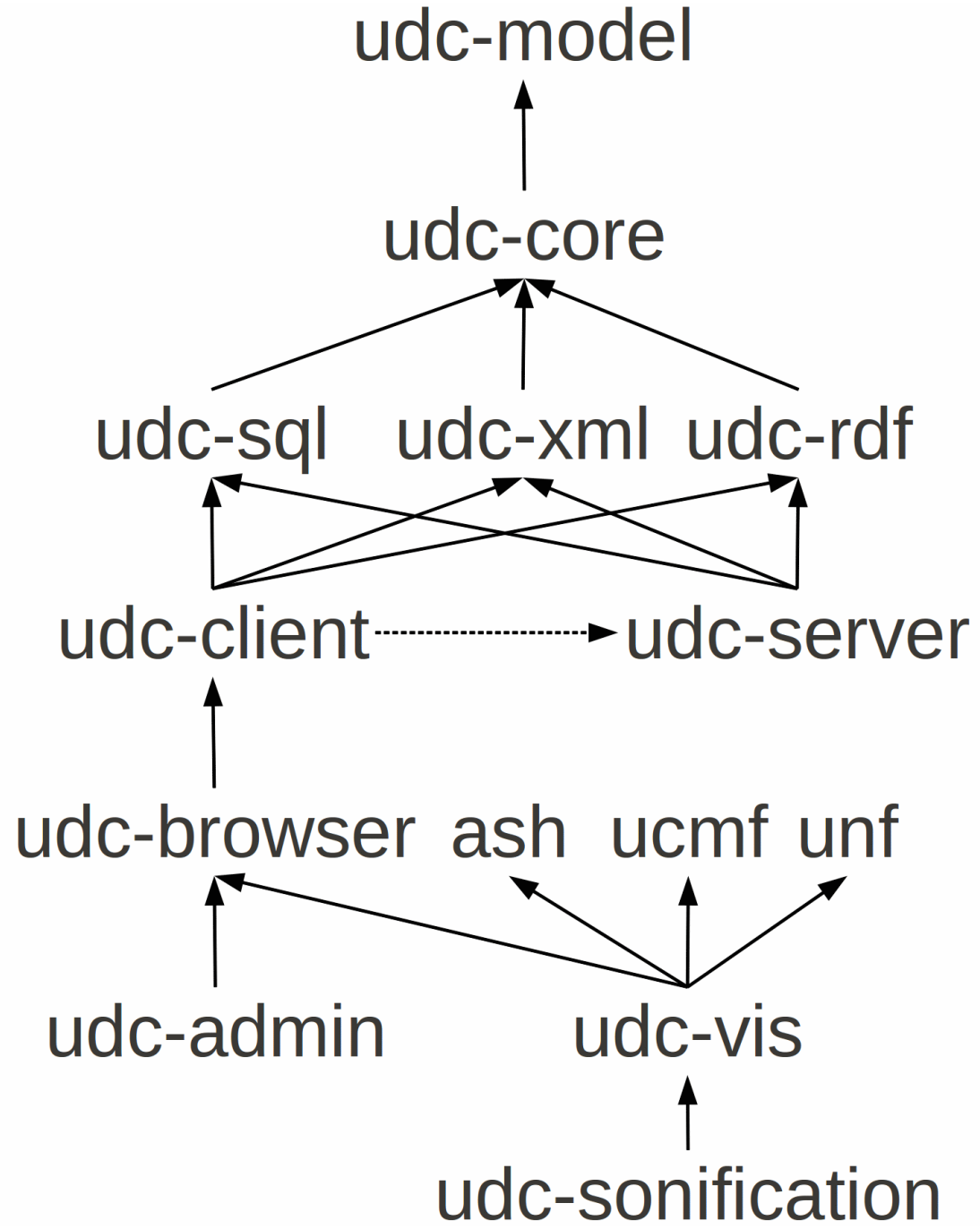




# Collaboration Idea

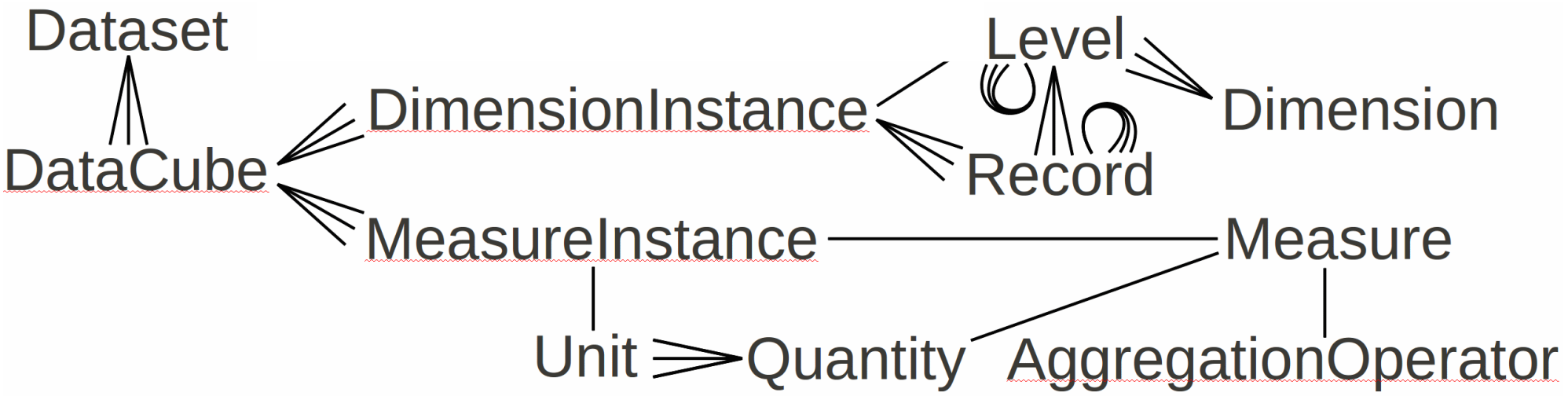
- Implement a UDC client which drives the data input for the hierarchical pie chart tool.

# Future Plans



The end.

# Knowledge


Dataset
DataCube
DimensionInstance
Level
Dimension
Record
MeasureInstance
Measure
Unit
Quantity
AggregationOperator