Managing Big Multidimensional Data: A Journey From Acquisition to Prescriptive Analytics

Torben Bach Pedersen
Daisy@CS@Aalborg University
Speaker Presentation

• I try to squeeze the world into cubes…

["Kitchen Funnel" by Donovan Govan licensed under CC BY-SA 3.0]

["Rubix cube.jpg" by Andromorfo licensed under CC BY-SA 4.0]
Agenda

• What is Big Data?
• What is Big Multidimensional Data?
  - And what is *really* new about it?
• What should we do differently?
  - Pretty much everything…😊

• How to handle volume, velocity, and variety?

• A new data cycle for Big Multidimensional Data
  - Merging steps
  - Hierarchical steps
  - Models in all steps
  - A new step: prescriptive analytics
Multidimensional Data

- **MD characteristics**
  - *Facts* (Sale)
  - *Dimensions* (Time, Product)
  - Facts form *cells* in MD *cubes*
  - Aggregatable *measures* (Price)
  - *Hierarchies* (Prod., Type, Categ.)

- **On-Line Analytical Processing (OLAP)**
  - Fast, interactive analysis of large amounts of data
  - Spreadsheets on steroids

- **Iterative queries of two types:**
  - Navigate/explore dimensions
  - Aggregate/disaggregate along dimensions (rollup/drilldown)

- **Traditionally used for** *business intelligence (BI)*
Multidimensional Data

- MD characteristics
  - Facts (Sale)
  - Dimensions (Time, Product)
  - Facts form cells in MD cubes
  - Aggregatable measures (Price)
  - Hierarchies (Prod., Type, Categ.)

- On-Line Analytical Processing (OLAP)
  - Fast, interactive analysis of large amounts of data
  - Spreadsheets on steroids

- Iterative queries of two types:
  - Navigate/explore dimensions
  - Aggregate/disaggregate along dimensions (rollup/drilldown)

- Traditionally used for business intelligence (BI)
Multidimensional Data

- **MD characteristics**
  - *Facts* (Sale)
  - *Dimensions* (Time, Product)
  - Facts form *cells* in MD *cubes*
  - Aggregatable *measures* (Price)
  - *Hierarchies* (Prod., Type, Categ.)

- **On-Line Analytical Processing (OLAP)**
  - Fast, interactive analysis of large amounts of data
  - Spreadsheets on steroids

- **Iterative queries of two types:**
  - Navigate/explore dimensions
  - Aggregate/disaggregate along dimensions (rollup/drilldown)

- **Traditionally used for business intelligence (BI)**
What is Business Intelligence?

- Business intelligence is “the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal”

- Business intelligence is “an umbrella term that includes the applications, infrastructure and tools, and best practices that enable access to and analysis of information to improve and optimize decisions and performance”
  - Gartner Reports, IT Glossary, 2013

- So, it’s about optimizing your business using data…

- For example:
  - Show the total sales by product category
  - What is the trend over time (drill-down by month)?
  - How do sales correlate with location (drill-down by store loc)?
A *Journey* of steps that data pass through in most BI applications
What is Big Data, then?

• "Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications."

• So, it should be so "big" that it becomes "difficult" to do it the traditional way…

• …We have to do things differently..
Big Data Characteristics

• "The 3 V’s" (but 1-2 V’s is “enough”)
• Volume
  ■ Very large data volumes
• Velocity
  ■ Data arrives very fast (data streams)
• Variety
  ■ Data has varied/complex formats/types/meanings

More V’s:
  ■ Veracity – how much can we trust data?
  ■ Viability – can our data be used for anything useful?
  ■ Visibility – data must be visible to the Big Data processes
  ■ Variability – the meaning of data changes over time/place/context
  ■ Visualization – complex visualization needed to fully understand
  ■ Value – what real value can this data add to our business?
BI Versus Big Data

- **Similarities (what is not so new?)**
  - Collecting, integrating, and analyzing data to gain knowledge
  - Large data volumes
  - Data (often) arrives at a fast pace

- **Differences (what is really new?)**

<table>
<thead>
<tr>
<th></th>
<th>BI</th>
<th>Big Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data types</td>
<td>Structured (mostly)</td>
<td>Unstructured (also)</td>
</tr>
<tr>
<td>Data sources</td>
<td>Mostly internal</td>
<td>Mostly external</td>
</tr>
<tr>
<td>Deployment</td>
<td>In-house</td>
<td>Cloud</td>
</tr>
<tr>
<td>History</td>
<td>Essential</td>
<td>(Often) less relevant</td>
</tr>
<tr>
<td>Users</td>
<td>Manager/controller</td>
<td>Data scientist</td>
</tr>
<tr>
<td>Precision</td>
<td>Exact results</td>
<td>Approximate results</td>
</tr>
<tr>
<td>Privacy</td>
<td>Not critical</td>
<td>Critical</td>
</tr>
<tr>
<td>Control over data</td>
<td>Almost full control</td>
<td>Little or no control</td>
</tr>
</tbody>
</table>

DaWaK, September 6-8, 2016
Illustrating The Change

- Image with girl drinking from a straw ("sipping data from a straw") and getting sprinkled with water in the face (removed due to unclear copyright)
(Typical) Types of Big Data

- **Search data**
  - Web pages, searches, rankings, etc.
  - Google’s data…the first type of Big Data
- **Social network data**
  - Updates from Twitter, Facebook, LinkedIn, user fora,…. Text, images, user info, Likes, location, friends-graph,…
- **Linked/Open Data**
  - Data shared/published on WWW, e.g., using Semantic Web techn.
- **But it is not just from WWW…**
- **Big Sensor Data**
  - Big Science Data (CERN Large Hadron Collider, etc.)
  - Big GPS/Location Data
  - Big RFID Data
  - Big Energy Data – the basis of the Smart Grid
Big Multidimensional Data

- Multidimensional characteristics
  - Facts, dimensions, hierarchies, measures, cubes,…
  - But different

- How to handle?

- Volume
  - …really big data volumes

- Velocity
  - …that arrive very fast

- Variety
  - …and has very different types/meanings?
A New Data Cycle

- New data cycle for Big Multidimensional Data

- **Merging** steps
  - Doing several steps in combination

- **Hierarchical** steps
  - Steps inside steps

- **Models** in all steps
  - Models for data acquisition, storage,…

- A new step: **prescriptive** analytics
  - Combining prediction and optimization
Volume – Typical Approach

• **Data parallelism**
  - Split data, compute in parallel, coordinate, redundancy
  - MapReduce/Hadoop
  - Lucene/Solr for text

• **Pros:**
  - Scalability, cheap HW, fault tolerant, (often) intuitive model

• **Cons:**
  - Load balancing, latency, (often) inefficient, low productivity
  - Work harder, not smarter 😊

[Hadoop Tutorial from Yahoo!](https://www.apache.org) by Yahoo! Inc. Licensed under [CC BY 3.0](https://creativecommons.org/licenses/by/3.0)
Volume: Efficiency

• Pure parallelization is not enough
• Efficient algorithms and data structures (still) necessary
• A particularly efficient data structure for multidimensional queries is *(compressed) bitmap indices*
  - So, what is that?
• Idea: make a ”position bitmap” for every possible value
  - #Danmark: 01110010101010… (row 2,3,4,7… has #Danmark)
  - #BigData: 10001101010101… (row 1,5,6,8… has #BigData)
  - Only takes (no. values)*(no. rows)*1 bit space
  - Very efficient ”index intersection” (CPU AND/OR) on bitmaps
• Problem: space usage
  - With \( m \) possible values and \( n \) rows: \( n \times m \) bits needed
  - But the probability of a 1 is only \( 1/m \) => very few 1’s
PLWAH [EDBT’10]

• Literal+fill words; split bitmaps into w-1 bit chunks
• 1 or more chunks with all 0’s/1’s = fill, otherwise literal

0

• Finally, merge fill words with "few bit” literals at the end

1

• Employs novel CPU instruction sets (POPCnt, etc)
• Storage: comparable to BBC (Oracle), half of WAH
• Speed: 40% faster than WAH, 15 times BBC (Oracle)
• US Patent, Algorhyme spin-out
An application: Algorhyme Query

- Oracle Data Cartridge
  - “DB Chip tuning set”
- AQ vs. Oracle Bitmaps
  - 10-15 times faster
- AQ vs. Oracle Text
  - 10-50 times faster
- AQ vs. Apache Lucene
  - 20-30 times faster
- Combined text and structured metadata
  - Up to 100 times faster

["Algorhyme Query" by Algorhyme A/S]
Ongoing Work

• Bitmap Indexing for Big Data
  ■ Compressed bitmap indexing for Apache Spark SQL

• Volume and Velocity
  ■ Both disk-based and in-memory data
  ■ Streaming input

• Variety
  ■ Categorical data
    ◆ Tweets with hash tags, …
    ◆ Item and set queries
  ■ Numerical data
    ◆ Sensor time series,…
    ◆ Range queries
Velocity: Typical Approach

- Everything in RAM
  - Avoid redundancy + disk intermediaries, recompute if necessary
- Apache Spark
  - Resilient Distributed Datasets (RDD’s)
  - Operators on RDD’s
- Pros
  - 10-100*faster
  - More productivity
- Cons
  - RAM expensive and limited
  - Standalone scenario
  - Misses some optimization potentials

Hadoop/MapReduce data sharing
Velocity: Typical Approach

• Everything in RAM
  ■ Avoid redundancy + disk intermediaries, recompute if necessary

• Apache Spark
  ■ Resilient Distributed Datasets (RDD’s)
  ■ Operators on RDD’s

• Pros
  ■ 10-100*faster
  ■ More productivity

• Cons
  ■ RAM expensive and limited
  ■ Standalone scenario
  ■ Misses some optimization potentials
Velocity: Fast Energy Data

- Many time series (supply, demand, flexibility,...)
- Data start out in the **future**
  - Long term forecast, (more accurate) medium term forecast, (even more accurate) short term forecast, more and more accurate
- And finally make it to the **present**
  - Read actual data value from sensor and store it (*inaccuracy/delay*)
- …and into the **past**
  - Keep for long term analytics and as basis for re-forecasting
- Key observation:
  - **Only** difference btw. forecasted and ”real” data is level of accuracy
- Idea
  - Use (better and better) *models* to represent **all** data
    - Past, present, future
  - *Model adaption* instead of loading (perhaps free 😊)
TimeTravel System [PVLDB’12]

- Past, future and combined (timetravel) queries
  - "Show average consumption for today and tomorrow"
- Exact queries (actual time series values)
  - "Show average consumption for today and tomorrow" (using detailed time series values)
  - Future values are (of course) not "exact" since they are forecasted
- Approximate queries (absolute/relative error)
  - "Show average consumption for today and tomorrow with up to 5% error"
  - Potential for huge performance gains
- Hierarchical model index
  - Chebyshev polynomials, progressively lower error
- Time series: Seasonal, Trend, Error components
  - Period hints for seasonality, e.g., 1 or 2 seasonalities per week
- PostgreSQL based prototype
- Up to 2 orders of magnitude smaller/faster
- Query past+future seamlessly with SQL!

 DST

Year

Week

Day

UK household power consumption

Seasonal

Trend

Error

DaWaK, September 6-8, 2016

24
TimeTravel Queries

• *(Past Query, Approximate) Show the 15-minutes power consumption for yesterday with an absolute max error of 100*
  
  \[
  \text{SELECT TIME, POWER\_CONSUMPTION FROM M\_UK [-96,0] PINTERVAL=15-MINUTE ERROR=100;}
  \]

• *(TimeTravel Query) Find daily maximum power consumption over the last and next week:*
  
  \[
  \text{SELECT MAX(Power\_Consumption) FROM M\_UK [-336,+336];}
  \]
TimeTravel Architecture

- **Building Module.** Hints+timeseries->hierarchical model index
- **Compression Module.** Reduce model storage by combining similar models
- **Query Processing Module.** Extends PostgreSQL processor/optimizer
  - Support approximate point range, aggregate and join queries
  - Traverses down the model index until required accuracy is reached.
- **Forecasting Module.** Predicts future time series values, estimates error and confidence, re-estimates forecast method parameters.
- **Maintenance Module.** Maintains hierarchical model with new time series values, adds new models to HMI or updates model parameters
The New Data Cycle

• How is TimeTravel different?
• Models in all steps
  - Same **models** for acquiring, storing, exploring, learning, predicting
• Merging steps
  - Acquiring, storing, and predicting **merged** (feedback loop)
  - Data acquisition through **model adaption**

![Diagram of data cycle](image)

Models (Chebyshev polynomials)
Some Really Big and Fast Data…

- Denmark is the world no. 1 in wind energy
  - Come and visit and you will feel why 😊
  - Vestas (top 2) and Siemens WP (no. 4) both DK-based
  - World record electricity from wind
    - 2015: 43%, december 2014: 110%

- Wind turbines
  - 500 sensors
  - 8 byte values sampled at 100Hz or more
  - 100+ turbines in a wind park
  - 100*100*500 = 5 million values/second = 40+ MB/sec
  - 40 MB * 3600 * 24 = 3.5+ TB/day = 1.3+ PB/year/park
  - They want to store 20+ years for 1000s of parks…

- Industry state of the art
  - 500 column SQL Server tables with 15 min averages…
Ongoing Work

- Model-Based Management of Big Sensor Data
  - Model-based load, storage, querying for Apache Spark SQL

- Key ideas
  - Massive correlation/redundancy in sensor streams
    - Over time and between sensors/turbines/parks/..
    - Can be exploited for massive compression
  - Approximate storage and querying
    - State the accuracy you actually need
  - Generic model-based storage
    - Library of models, auto-pick to fit data
  - Streaming input
    - Build models on the fly

- New Data Cycle
  - Models in all steps, merging steps
Variety: Big RFID Data

• "BagTrack – styg på bagagen"
  - Daisy, Lyngsoe, SAS (Arlanda!), IATA, AAL - app
  - Bag tags w. RFID, license plate (ID), route, date
  - Vision: real-time world-wide baggage info in 2020: 50% less baggage problems, save 1.2 bio. US$/year

• Daisy Big Data research
  - Real-time data and queries
  - OLAP/DW – analyze processes and measurements
  - Data mining: problems/causes in event sequences
  - Big/complex data, 1000+ airports
  - Data cleansing – get true meaning from RFID reads
    - Learning-Based Cleansing for Indoor RFID Data [SIGMOD’16]
RFID Based Indoor Positioning

• Proximity analysis
  - An RFID reader detects an RFID tag when the tag (the object with the tag) enters the reader’s detection range
  - Deployment locations of RFID readers are recorded in advance.
  - An example from Aalborg Airport, Denmark

• Raw reading format
  - (objectID, readerID, t)

• Such raw data contains two types of errors
  - False positive
    - Cross readings
  - False negative
    - Missing readings
False Positives

- A reader mistakenly reads out the tags which are outside its intended detection range
  - Possible causes: metal reflection, antenna re-direction, etc.
False Negatives

- A reader fails to read out a tag that is actually in its intended detection range
  - Possible causes: power loss, tag orientation, distractors, etc.

Problem

Cleansing Indoor RFID Tracking Data
- To *reduce* false positives
- To *recover* false negatives
Approach Overview

• Data transformation
  - Raw (unlabeled) readings input: (tagID, readerID, t)
  - Binary vector representation output: \( V_r^{(0)}, V_r^{(1)}, \ldots, V_r^{(T-1)} \in \{0, 1\}^M \)
    - \( T \): discretized timestamp numbers
    - \( M \): the number of RFID readers

<table>
<thead>
<tr>
<th>tagID</th>
<th>readerID</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_0 )</td>
<td>( R_1 )</td>
<td>( t_1 )</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>( R_1 )</td>
<td>( t_3 )</td>
</tr>
<tr>
<td>( t_4 )</td>
<td>( R_4 )</td>
<td>( t_7 )</td>
</tr>
<tr>
<td>( t_8 )</td>
<td>( R_4 )</td>
<td>( t_9 )</td>
</tr>
</tbody>
</table>

• Learning-based cleansing

Hidden Markov Model (HMM) with parameters learned from raw data
Indoor RFID Multi-variate HMM

- Multi-variate HMM $\lambda = (S, O, A, B, \pi)$
  - $S = \{s_1, \ldots, s_N\}$ is a set of hidden states
  - $O = \{0, 1\}^M$ is a set of possible observations
  - $A$: $N \times N$ transition probability matrix for hidden states
  - $B$: $N \times M$ observation probability matrix
    - $b_{ih} (i=1, \ldots, N; h=1, \ldots, M)$ is the probability of having a reading by reader $h$ when the hidden process is in state $s_i$
  - $\pi$: $N$-dimensional initial state probability vector

- Probabilistic parameters are learned from the raw data
- What about cardinality and structure of the state space?
  - Which topology (no. states+connections)?
State Space Design

Minimum State Model (MSM)
- A state for each reader
- A state for each hallway
- A state for each indoor partition connected with >1 reader

Last State Model (LSM)
- Two states for each reader
- One state indicates that an object is in a reader’s range
- Another state represents that the reader is the last one that has detected the object

In-between State Model (ISM)
- A state for each reader
- A state for each pair of adjacent readers

The designs may be applied to different indoor layouts and movements (directional or random).
Learning and Cleansing

• Learning HMM parameters
  - Using a standard EM (Expectation Maximization) algorithm
  - Input
    • Uncleansed transformed data (raw/unlabeled)
    • One of the state space designs
  - Output
    • A new model $\lambda$ with parameters learned from raw data

• Data cleansing
  - Input: raw RFID transformed to binary observation sequence
  - The standard Viterbi Algorithm computes the most probable hidden state sequence $\hat{s}$
  - The most probable observation sequence is determined by maximizing the conditional probability under $\hat{s}$, i.e.,
    $$P(V^{(t)}_r = v^{(t)}_r \mid S^{(t)} = \hat{s}^{(t)})$$
Experiments

• Approaches in comparison
  - Our learning-based approach: MSM, LSM, ISM
  - Graph-based approach: GRAPH [1,2]
    - Requires detailed domain knowledge (distance between readers, moving speed of objects, sampling rate, etc.)
  - Conditional Trajectory Graph approach: CTG [7]
    - Even more prior knowledge ((in)reachability, travel time, latency,..)

• Main results
  - Gets better with more data
  - Given enough data, our approach beats GRAPH+CTG
    - With just unlabeled data and much less prior knowledge

• Take home message
  - Data+learning beats rule-based hand-crafted models
  - Easy deployment
The New Data Cycle

• How is IR-MHMM different?
• Models in all steps
  - Models also in cleansing
• Hierarchical steps
  - Full data cycle inside cleansing
Prescriptive Analytics: A New Step

Acquiring → Cleansing → Storing → Exploring → Learning (models) → Predicting → Prescribing

Idea: Prescribe the right course of action to optimize a given goal
What is Prescriptive Analytics?

Descriptive Analytics: What happened?

Predictive Analytics: What will happen?

Prescriptive Analytics: How to make it happen?

Foresight

Insight

Hindsight

INTELLIGENCE

DIFFICULTY

INFORMATION

OPTIMIZATION
An Example from Smart Energy

- Balancing demand and supply

Step 1: Forecast RES production

Step 2: Generate demand FlexOffers

Step 3: Optimize schedules

Simplified problem, the real problem also involves time flexibility
Traditional Solution

• Implementation based on RDBMS + R + Java solver

1. Bash script (12 lines)
2. Forecasting program in R (53 lines)
3. Scheduling program in Java (116 lines)
4. R (lm)
5. Postgres- SQL
6. Java solver (3394 lines)

• Problems:
  - Non-integrated specialized glued together in ad-hoc fashion
  - Labor-intensive, error-prone, inefficient

```bash
#!/bin/bash
export mirabelDbUrl="jdbc:postgresql://localhost/postgres";
export mirabelDbUsername="postgres";
export mirabelDbPassword="";
export mirabelRoot="pwd";
export CLASSPATH="$CLASSPATH:../FlexOffers/Implementation/SDB-
scheduling-experiments/target/sdb-scheduling-experiments-0.9.9-
SNAPSHOT.jar:../FlexOffers/Implementation/SDB-scheduling-
experiments/target/dependency/*"

# Run forecasting
(time Rscript ergv_forecast.R) &> output_fc_r.txt

# Check for the forecasting error
fcerror=`psql -d postgres -c "SELECT * FROM forecast_error"
-Ptuples_only`

echo "Forecasting error (MAPE): " $fcerror

# Run scheduling
(time java -Xmx1000m -Xss10m
org.mirabel.aggregation.experiments.AggSchExperiment) &>
output_sch_c.txt

# Check for imbalance
schImb=`psql -d postgres -c "SELECT * FROM scheduling_imbalances;"
-Ptuples_only`

echo "Scheduling remaining imbalance (kWh): " $schImb
```
SolveDB Solution [SSDBM’16]

- **SolveDB (PostgreSQL + solvers)**
  - Integrates RDBMS and LP/MIP, BB, and specialized solvers
  - Uses SQL extension for problems

  - Lines of code: 3571 versus 237
  - I/O time: 7.3 versus 0.8 secs

```
SOLVESELECT sch IN (SELECT fo, NULL::schedule AS sch
FROM flexobject) AS t
SUBJECTTO (SELECT is_instanceof(sch, fo) FROM t),
(SOLVESELECT load IN (SELECT time, load
FROM hist_load) AS s
SUBJECTTO (SELECT time, temp
FROM temp_data)
WITH solverFO)
WITH solverFS(rndseed:=12345, sn:=3176)
```

Specifies forecasting

Specifies scheduling
Optimization problems

- Many real-world problems are optimization problems

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad g_i(x) \leq 0, \quad i = 1, \ldots, m \\
& \quad h_i(x) = 0, \quad i = 1, \ldots, p
\end{align*}
\]

- Energy Planning
- Finance
- Management
- Product Design
- …
Optimization in real-world applications

Database

Problem

\[
\begin{align*}
\text{minimize} & \quad f(x) = 0.4(x_1/x_7)^{0.67} + 0.4(x_2/x_8)^{0.67} + 10 - x_1 - x_2 \\
\text{subject to} & \quad 1 - 0.0588x_5x_7 - 0.1x_1 \geq 0 \\
& \quad 1 - 0.0588x_6x_8 - 0.1x_1 - 0.1x_2 \geq 0 \\
& \quad 1 - 4x_3/x_5 - 2/((x_3^{0.71}x_5)^{1.3}) - 0.0588x_7/x_3^{1.3} \geq 0 \\
& \quad 1 - 4x_4/x_6 - 2/((x_4^{0.71}x_6)^{1.3}) - 0.0588x_8/x_4^{1.3} \geq 0 \\
& \quad 0.1 < f(x) < 4.2 \\
& \quad 0.1 \leq x_i \leq 10, \quad i = 1, 2, \ldots, 8
\end{align*}
\]

Solution

\[
\begin{align*}
x_1 &= 6.46511 \\
x_2 &= 2.23271 \\
x_3 &= 0.66740 \\
x_4 &= 0.59576 \\
x_5 &= 5.93268 \\
x_6 &= 5.52724 \\
x_7 &= 1.01332 \\
x_8 &= 0.40067
\end{align*}
\]

E.G., ENERGY PLANNING

- Demand measurements
- Supply flexibility data

Optimize forecast model parameters
Forecast demand values
Optimize supply schedule
Store and analyze the schedule

Optimization and database querying are interleaved
Common in prescriptive analytics applications
Traditional v.s. SolveDB approach

- Two software packages
- Error-prone
- Two languages
- I/O overhead
- Difficult to version

+ A single system
+ A single language (SQL-like)
+ I/O is reduced
SolveDB: Use-case **0/1** Knapsack

**Input relation**

<table>
<thead>
<tr>
<th>item</th>
<th>weight</th>
<th>profit</th>
<th>quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>item 1</td>
<td>10.0</td>
<td>5.0</td>
<td>NULL</td>
</tr>
<tr>
<td>item 2</td>
<td>9.0</td>
<td>4.5</td>
<td>NULL</td>
</tr>
<tr>
<td>item 3</td>
<td>1.5</td>
<td>2.0</td>
<td>NULL</td>
</tr>
<tr>
<td>item 4</td>
<td>7.0</td>
<td>3.0</td>
<td>NULL</td>
</tr>
</tbody>
</table>

**Output relation**

<table>
<thead>
<tr>
<th>item</th>
<th>weight</th>
<th>profit</th>
<th>quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>item 1</td>
<td>10.0</td>
<td>5.0</td>
<td>1</td>
</tr>
<tr>
<td>item 2</td>
<td>9.0</td>
<td>4.5</td>
<td>0</td>
</tr>
<tr>
<td>item 3</td>
<td>1.5</td>
<td>2.0</td>
<td>1</td>
</tr>
<tr>
<td>item 4</td>
<td>7.0</td>
<td>3.0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Pick items, so that:
  1. **Total weight** does not exceed 15
  2. **Profit** is maximized

- Can be included to other SQL queries, e.g., INSERT, UPDATE

```sql
SOLVESELECT quantity IN (SELECT * FROM items) AS u
  MAXIMIZE (SELECT SUM(quantity*profit) FROM u)
SUBJECTTO (SELECT SUM(quantity*weight)<=15 FROM u),
  (SELECT 0<=quantity<=1 FROM u)
WITH solverlp.mip();
```
SolveDB: Solve query execution

User query

Parse SOLVESELECT

Find the most suitable view solver

Initialize and inline a solver workflow

Contains SOLVESELECT?

Yes

Optimize the whole query

Execute the query

Query results
SOLVESELECT syntax

List of attributes with decision (unknown) variables
- Different data types are supported
- May include initial values

Specify the input relation
- Specified by any SELECT

Specify objective functions to be minimized/maximized

Specify constraints and/or data

SOLVESELECT `col_name [, ...]` IN ( `select_stmt` ) [AS alias]
[MINIMIZE ( `select_stmt` )] [MAXIMIZE ( `select_stmt` )] |
MAXIMIZE ( `select_stmt` ) [MINIMIZE ( `select_stmt` )] ]
[SUBJECTTO ( `select_stmt` ) [, ...]]
[WITH solver_name [. . .] [( param[:= expr] [, . . .] )]]
SolveDB: Solve query execution

User query

Parse SOLVESELECT

Find the most suitable view solver

Initialize and inline a solver workflow

Contains SOLVESELECT?

Yes

Optimize the whole query

Execute the query

Query results

No
SolveDB: Solver types

Problem descriptor $d$
Configuration parameters $p$

SolveDB solver

Problem solution $s$

View Solvers
- Composite View Solvers
- Atomic View Solvers

Relational Solvers

Physical Solvers
- e.g., CPLEX, GLPK

Activated at solve query parse time
Activated at solve query execution time

SolveDB-specific solvers (novel)
SolveDB: View solvers

\[
\text{SOLVESELECT} \quad \text{quantity} \quad \text{IN} \quad (\text{SELECT} \quad \ast \quad \text{FROM} \quad \text{items}) \quad \text{AS} \quad u \\
\text{MAXIMIZE} \quad (\text{SELECT} \quad \text{SUM}(\text{quantity} \ast \text{profit}) \quad \text{FROM} \quad u) \\
\text{SUBJECTTO} \quad (\text{SELECT} \quad \text{SUM}(\text{quantity} \ast \text{weight}) \leq 15 \quad \text{FROM} \quad u), \\
(\text{SELECT} \quad 0 \leq \text{quantity} \leq 1 \quad \text{FROM} \quad u) \\
\text{WITH} \quad \text{solverlp.mip} \quad (\text{presolve} := \text{true});
\]

\[d_v=(R_{in}, U, V)\]

View Solvers

\[p_v\]

Composite View Solvers

Atomic View Solvers

Produce another SOLVESELECT query

Produce a query to invoke the relational solver

\[s_v \quad \text{a query (plan) to produce} \quad R_{out}\]
SolveDB: Solve query execution

User query → Parse SOLVESELECT → Find the most suitable view solver → Initialize and inline a solver workflow

Contains SOLVESELECT? (Yes/No)

No → Optimize the whole query → Execute the query → Query results

Yes → Optimize the whole query
SolveDB: Composite view solvers

QUERY REWRITE

\[
\text{SOLVESELECT} \quad \text{quantity IN (SELECT * FROM items)} \\
\text{WITH solver_knapsack());}
\]

Rewrites to

\[
\text{SOLVESELECT} \quad \text{quantity IN (SELECT * FROM items) AS u} \\
\text{MAXIMIZE} \quad (\text{SELECT SUM(quantity*profit) FROM u}) \\
\text{SUBJECTTO} \quad // \text{Inherent constraints} \\
(\text{SELECT SUM(quantity*weight)\leq15 FROM u}), \\
(\text{SELECT 0\leqquantity\leq1 FROM u}) \\
\text{WITH solverlp.mip());}
\]
SolveDB: Composite view solvers

**QUERY REWRITE**

SOLVESELECT quantity IN (SELECT * FROM items) AS u
SUBJECTTO (SELECT sum(quantity) > 2 FROM u)
WITH solver_knapsack();

Rewrites to

SOLVESELECT quantity IN (SELECT * FROM items) AS u
MAXIMIZE (SELECT SUM(quantity*profit) FROM u)
SUBJECTTO  // Inherent constraints
(SELECT SUM(quantity*weight) <= 15 FROM u),
(SELECT 0 <= quantity <= 1 FROM u),
// User (external) constraints
(SELECT sum(quantity) > 2 FROM u)
WITH solverlp.mip();
SolveDB: Atomic view solvers

INVOKE RELATIONAL SOLVER UDF

Relational solver invoked

Solution table

<table>
<thead>
<tr>
<th>order</th>
<th>item</th>
<th>weight</th>
<th>profit</th>
<th>quantity: LpExp</th>
<th>var_nr</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>item 1</td>
<td>10.0</td>
<td>5.0</td>
<td>NULL(1[1])</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>item 2</td>
<td>9.0</td>
<td>4.5</td>
<td>NULL(1[2])</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>item 3</td>
<td>1.5</td>
<td>2.0</td>
<td>NULL(1[3])</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>item 4</td>
<td>7.0</td>
<td>3.0</td>
<td>NULL(1[4])</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

SELECT SUM(quantity*profit) FROM u

<table>
<thead>
<tr>
<th>column 1: LpExp</th>
</tr>
</thead>
</table>

SELECT SUM(quantity*weight) <= 15 FROM u

<table>
<thead>
<tr>
<th>column 1: LpCtr</th>
</tr>
</thead>
</table>

SELECT 0 <= quantity <= 1 FROM u

<table>
<thead>
<tr>
<th>column 1: LpWeight</th>
<th>profit</th>
<th>quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1[1]&lt;='0)</td>
<td>0</td>
<td>5.0</td>
</tr>
<tr>
<td>(1[2]&lt;='1)</td>
<td>9.0</td>
<td>4.5</td>
</tr>
<tr>
<td>(1[3]&lt;='0)</td>
<td>1.5</td>
<td>2.0</td>
</tr>
<tr>
<td>(1[4]&lt;='1)</td>
<td>7.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>
SolveDB: Solve query execution

User query

Parse SOLVESELECT

Find the most suitable view solver

Initialize and inline a solver workflow

Contains SOLVESELECT?

Yes

Optimize the whole query

Execute the query

Query results

No
SolveDB: Solve query optimization

- Solver design optimizations
- Solver runtime optimizations
- 1-time queries
- Repeated queries

- Order-level view ($R^o_{in}$) materialization (OMat)
- Efficient solver UDTs and UDFs (OptStruct)
- Model-level view ($R^m_{in}$) parametrization (Param)
- Solver Advisor (Advice)
- Problem partitioning (Partit)
- Built-in solver optimization (Built-in)
- Solution materialization (SMat)
- Meta-optimization (Meta)

Descriptor $d_v$

<table>
<thead>
<tr>
<th>One-time, directly</th>
<th>Advice</th>
<th>OMat</th>
<th>OptStruct</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-time, iteratively</td>
<td>Advice</td>
<td>OMat</td>
<td>OptStruct</td>
</tr>
<tr>
<td>Repeated, directly</td>
<td>Advice, SMat</td>
<td>OMat</td>
<td>OptStruct</td>
</tr>
<tr>
<td>Repeated, iteratively</td>
<td>Advice, SMat</td>
<td>OMat</td>
<td>OptStruct</td>
</tr>
</tbody>
</table>

Sub-problem count

- Sub-total time [s] vs. Tuple count $P_{Q2}$

Graph showing the relation between sub-total time, tuple count, and solver performance.
SolveDB: Solve query execution

User query

Parse SOLVESELECT

Find the most suitable view solver

Initialize and inline a solver workflow

Contains SOLVESELECT?

Yes

Optimize the whole query

No

Execute the query

Query results

Executed like any other database query
SolveDB: Experimental Results

• SolveDB v.s. GLPSOL v.s. R
  - 1.3-2.5x less code
  - 2.2-18x reduced I/O time
  - 2.6x less total time

• SolveDB v.s. C++ v.s. R
  - Much less code (up to orders of magnitude)
  - SolveDB performs 30-100 times better than R
  - C++ performs slightly better, but requires much more code

• SolveDB v.s. Logicblox v.s. Tiresias
  - Significantly less code in all cases
  - Extended SQL versus SQL+Datalog
  - I/O time is >100x smaller
The New Data Cycle

- Merging steps
  - All steps done in SolveDB
- A new step: **prescriptive** analytics
  - Integrated support directly in extended SQL
- Models in all steps
  - Also prescriptive models
Data-intensive CPS: More Models

Integrate models of processes, data, and users in energy, transport,…

DaWaK, September 6-8, 2016

[Original slide by Kim G. Larsen]
Big Multidimensional Data

• Volume (PLWAH)
• Velocity (TimeTravel)
• Variety (RFID and energy data)

A new data cycle for Big Multidimensional Data

- Merging steps
  - TimeTravel, IR-MHMM, SolveDB
- Hierarchical steps
  - IR-MHMM
- Models in all steps
  - TimeTravel, IR-MHMM, SolveDB
- A new step: prescriptive analytics
  - SolveDB

Bad news: Big Multidimensional Data is harder 😞
Good news: good job security for data geeks😊
Key References

• http://people.cs.aau.dk/~tbp
• Pedersen, Managing Complex Multidimensional Data, eBISS 2012, Springer LNBIB
• Deliège et al. Position list word aligned hybrid: optimizing space and performance for compressed bitmaps. EDBT 2010
• Algorhyme: www.algorhyme.com
• Khalefa et al. Model-based Integration of Past & Future in TimeTravel. PVLDB 5(12), 2012
• Baba et al. Learning-Based Cleansing for Indoor RFID Data. SIGMOD 2016
• Siksnys et al. SolveDB: Integrating Optimization Problem Solvers Into SQL Databases, SSDBM 2016
• http://daisy.aau.dk/solvedb
• DiCyPS project: www.dicyps.dk
Acknowledgements

- Many slides borrowed from colleagues and collaborators
  - Asif I Baba
  - Dalia Kaulakiene
  - Davide Frazzetto
  - Francois Deliege
  - Kim G. Larsen
  - Laurynas Syksnys
  - Mohamed Khalefa