14. Wrap Up
Course evaluation
<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The lecturer explained the subject understandably and clearly</td>
<td>Not true</td>
<td></td>
<td></td>
<td></td>
<td>Absolutely true</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>The lecturer clarified what I should be learning in this course unit (learning goals) and returned to this point regularly</td>
<td>Not true</td>
<td></td>
<td></td>
<td></td>
<td>Absolutely true</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>The lecturer made the significance of this lecture clear</td>
<td>Not true</td>
<td></td>
<td></td>
<td></td>
<td>Absolutely true</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>The lecturer motivated me to take an active part in the course</td>
<td>Not true</td>
<td></td>
<td></td>
<td></td>
<td>Absolutely true</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>The materials made available (e.g. lecture notes, textbook, handouts, etc.) helped me to understand and address the course content</td>
<td>Not true</td>
<td></td>
<td></td>
<td></td>
<td>Absolutely true</td>
<td></td>
</tr>
</tbody>
</table>

n=75  av.=4.6  md=5.0  dev.=0.6
n=74  av.=4.4  md=5.0  dev.=0.8
n=75  av.=4.6  md=5.0  dev.=0.6
n=75  av.=4.5  md=5.0  dev.=0.7
n=75  av.=4.4  md=5.0  dev.=0.8
## Exercises

1. The exercises helped me to understand and apply the content of the lecture
   - Not true
   - Absolutely true

2. The exercises were supervised helpfully by the Assistants
   - Not true
   - Absolutely true

---

<table>
<thead>
<tr>
<th>n</th>
<th>av.</th>
<th>md</th>
<th>dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>69</td>
<td>4.4</td>
<td>5.0</td>
<td>0.9</td>
</tr>
<tr>
<td>68</td>
<td>4.2</td>
<td>5.0</td>
<td>1.1</td>
</tr>
</tbody>
</table>
Motivation and Learning

8.1) I attended the lecture because I am interested in the subject

8.2) I attended the lecture because it is compulsory

8.3) I attended the lecture because of the lecturer

8.4) I am able to explain the most important material learned in this course unit clearly and understandably to a younger student

n=75  av.=4.5  md.=5.0  dev.=0.7
n=74  av.=1.5  md.=1.0  dev.=1.1
n=72  av.=2.8  md.=3.0  dev.=1.5
n=77  av.=3.9  md.=4.0  dev.=1.0
General satisfaction

9.1) How satisfied were you in general with the course unit?

Very unsatisfied

Very satisfied

n=75  av.=4.4  md.=5.0  dev.=0.8
### D-INFK questions

<table>
<thead>
<tr>
<th>Question</th>
<th>Rating</th>
<th>Count</th>
<th>Average</th>
<th>Median</th>
<th>Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>In comparison to other courses in the same course category, the specialist knowledge required was</td>
<td>much less</td>
<td>71</td>
<td>2.9</td>
<td>3.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>much less</td>
<td>69</td>
<td>3.0</td>
<td>3.0</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>Not true</td>
<td>78</td>
<td>4.1</td>
<td>4.0</td>
<td>0.9</td>
</tr>
</tbody>
</table>
The questionnaire enabled me to express my opinion of this course unit sufficiently

Not true | | | Absolutely true

n=76   av.=4.3   md=4.0   dev.=0.8
Constructive remarks: content

Data in the large
(Hadoop, ...)

Data in the small
(XML, JSON...)

vs.
Constructive remarks: projects

Reintroduce a big project?

More competitions?
Constructive remarks: clicker app well received

What is index-free adjacency?

- Instead of joins, the data is structured and linked using native pointers in memory.
- The adjacency matrix is indexed on rows and columns for efficient traversal.
- Relational joins are used instead of adjacency matrices.
- It is a technique with which documents are sorted along a given field, which allows binary search instead of an index lookup.
Constructive remarks: material

Not enough material? Slides with a bit more text? Script?

Too much material? Select more?
Constructive remarks: data center

In January because no resources left in December
Constructive remarks: time
10 Design Principles of Big Data
1. Learn from the past
Tables won't disappear

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Shapes matter!


2. Keep the design simple
A Wisdom

Everything Should Be Made as Simple as Possible, But Not Simpler.

- Albert Einstein
3. Modularize the architecture
Each data shape has one (or several)...

Paradigm
Data model
Syntax
Query language
Architecture
Mapping to other shapes
# Data models

<table>
<thead>
<tr>
<th>Everything is a...</th>
<th>So you...</th>
</tr>
</thead>
<tbody>
<tr>
<td>cube</td>
<td>slice and dice</td>
</tr>
<tr>
<td>table</td>
<td>select, project, group, join</td>
</tr>
<tr>
<td>tree</td>
<td>index, lookup, navigate</td>
</tr>
<tr>
<td>graph</td>
<td>traverse, find patterns</td>
</tr>
<tr>
<td>bunch of text</td>
<td>filter, map, group, aggregate</td>
</tr>
</tbody>
</table>
## Syntaxes

<table>
<thead>
<tr>
<th>If you have a...</th>
<th>You can serialize it as...</th>
</tr>
</thead>
<tbody>
<tr>
<td>cube</td>
<td>XBRL (CSV)</td>
</tr>
<tr>
<td>table</td>
<td>CSV</td>
</tr>
<tr>
<td>tree</td>
<td>XML, JSON, BSON, Protocol buffers</td>
</tr>
<tr>
<td>graph</td>
<td>RDF</td>
</tr>
<tr>
<td>bunch of text</td>
<td>as is</td>
</tr>
</tbody>
</table>
## Query languages

<table>
<thead>
<tr>
<th>If you have a...</th>
<th>You can query it with...</th>
</tr>
</thead>
<tbody>
<tr>
<td>cube</td>
<td>MDX</td>
</tr>
<tr>
<td>table</td>
<td>SQL</td>
</tr>
<tr>
<td>tree</td>
<td>XQuery, N1QL, UNQL, JSONiq, ...</td>
</tr>
<tr>
<td>graph</td>
<td>SPARQL, Cypher, ...</td>
</tr>
<tr>
<td>bunch of text</td>
<td>your own code (with MapReduce, Spark)</td>
</tr>
</tbody>
</table>
4. Homogeneity in the large
The progress made (1956-2010): Logarithmic

- **Capacity**: 622,100,131x
- **Throughput**: 11,719x
- **Latency**: 8x

5. Heterogeneity in the small
6. Separate metadata from data
7. Abstract logical model from its physical implementation
## Mappings

<table>
<thead>
<tr>
<th>If you have a...</th>
<th>You can easily store it as a...</th>
</tr>
</thead>
<tbody>
<tr>
<td>cube</td>
<td>table</td>
</tr>
<tr>
<td>table</td>
<td>tree</td>
</tr>
<tr>
<td>tree</td>
<td>graph</td>
</tr>
<tr>
<td>graph</td>
<td>table</td>
</tr>
<tr>
<td>bunch of text</td>
<td>table (set of key-value pairs)</td>
</tr>
</tbody>
</table>
Architecture evolution

Fancy declarative language

MapReduce/Spark

Hidden storage
8. Shard the data
9. Replicate the data
10. Buy lots of cheap hardware
Master-slave architecture
Design choices
You may have to...

1. Learn an existing product
2. Choose which product to use
3. Design a whole new technology or product
What's the shape of my data?


How much data do I need to store?
What do I want to do with my data?
Do I really need to reinvent the wheel?

Really?

Yes, really?
Data in 2030?
1. MapReduce and Spark (or the like) will be invisible
2. Data shapes will matter more than ever (and new shapes will be discovered?)
3. Querying all data shapes will be standardized (and maybe even unified?)
4. UIs will be the norm for end users (maybe even superseded by ML and AI)
5. Data will be fully fungible
(Web of Data 2.0)
6. Data and money will have converged
(Data will be an investment asset class)
Exam
Material

Slides

Recordings

Textbooks

Theoretical exercises

Practical exercises

Clicker questions
Let's start training on the exam!