Modern Database Systems

Seminar

Summer Term 2026

Organization

Course Goals

Primary goals:

- Learn how to write a scientific text
- Learn how to give a scientific talk
- Structure ideas from multiple research papers

Secondary goals:

- Practice to read scientific papers
- Understand modern cloud database system architectures
- Have interesting discussions with your peers

Deliverables

During the course each of you will create:

- A report proposal that summarizes your reports topic and focus ($<$1 page)
- A research report **draft** (5-8 pages + references)
- 2 peer reviews for your fellow students ($<$1 page each)
- A research report (5-8 pages + references)
- A pre-session protocol that shows your preparation for the presentation. ($<$1 page)
- A presentation (13-15 minutes)
- Meaningful contribution to the panel discussion in your presentations session

Grading

Rough estimate of grading contributions:

- $\approx 45\%$ Report
- $\approx 25\%$ Presentation
- $\approx 10\%$ Report Proposal
- $\approx 10\%$ Peer reviews
- $\approx 10\%$ Pre-session protocol and panel discussion

This is subject to change!

Timeline

Preliminary timeline:

- Mo 20.04.2026 introduction lecture 01
- Mo 27.04.2026 introduction lecture 02 | **submit** topic preferences
- Mo 04.05.2026 introduction lecture 03
- Mo 11.05.2026 $\emptyset$ | **submit** report proposal
- Mo 18.05.2026 $\emptyset$
- ~~Mo 25.05.2026~~ Pfingstferien
- Mo 01.06.2026 $\emptyset$
- Mo 08.06.2026 $\emptyset$
- Mo 15.06.2026 $\emptyset$ | **submit** report draft
- Mo 22.06.2026 presentation session 01
- Mo 29.06.2026 presentation session 02 | **submit** peer reviews
- Mo 06.07.2026 presentation session 03
- Mo 13.07.2026 presentation session 04 | **submit** final report
- **Attendance to all sessions in presence is mandatory**

Contact

- Website with slides: http://db.in.tum.de/teaching/ss26/seminarModernDatabaseSystems
- Gitlab with topics and schedule: https://gitlab.db.in.tum.de/Rieger/mdbs26
- Join our Mattermost Team MDBS26
- Please ask questions in the Discussion Channel
- Christoph Rotte: @ge93lej
- Maximilian Rieger: @rieger

Requirements for Deliverables

- The following slides cover our criteria for the individual deliverables.
- You may use them as a reference when starting to work on each piece.

Report Requirements (1) Structure

You will get a session topic and a focus topic assigned. We expect you to follow this structure:

- **Abstract**: Summarize all contents and results of your report.
- **Introduction**: Show the relevance of the problem and where it occurs. List what your report addresses.
- **Background**: Explain required previous knowledge. (optional)
- **Main Part**: Describe core ideas, methods, and solutions of your topic. This might include more than one section.
- **Related Work**: Find relevant work on the same and similar topics. What has been done before? What has been built on top of this later? Describe how your presented method differs from these others.
- **Research Idea**: Describe an interesting path for further research on this topic. Maybe for your own thesis?
- **Conclusion**: Briefly recap the contents and value of your report. Show the impact of your presented method.

Try to add value to your work. There should be a reason to read your report instead of the original papers.

Report Requirements (2) Further Points

Furthermore, the following points are important for your report:

- **Goal**: Choose one goal for the content of your report. All parts of your text should contribute to that goal. Possible goals might be:
  - Show that problem XY should be approached with solution AB.
  - Compare two or more approaches/systems and show the advantages, disadvantages, and use-cases.
  - Explain how approach/system XY works in detail and highlight relevant approaches and optimizations.
- **Content**: You should write about your session topic. The scope of your report should exceed the scope of your focus topic papers.
- **Language**: Use good scientific writing style. Write precise, concise, clear, and uncomplicated.
- **No Plagiarism**: Write the report on your own. If we notice that your report includes any text you did not understand or find long sections that just paraphrase sources you will not pass this course.

Report Requirements (3) Further Points

Furthermore, the following points are important for your report:

- **References**: Show your work on literature review by citing relevant sources. Adhere to citation guidelines.
- **Figures**: Your report needs to include at least one figure that actually enriches the report. Good are visualizations explaining algorithms, examples, architecture overviews. You may re-use external figures (although, creating your own can be very useful). Properly cite figures that are not your own. Caption text of all figures has to be your own.
- **Length**: Stay within a page count of 5-8 pages. References do not count.
- **Format**: Your report needs to use our template. If there is *any* visual difference to this template, we will deduct points. You have to submit as PDF.

Report Draft Requirements

Submit an earlier version of your final report for the peer review process.

- **Content**: This should be as close to your final submission as possible. It should be possible for your peers to give you meaningful feedback in their reviews.
- **Length**: The same as the final report.
- **Format**: The same as the final report.

Peer Review Requirements

You will read two of your peers reports and write reviews for them. We will follow a single-blind review process: Reviewers stay anonymous.

- **Content**: Pay attention to the following points:
  - **Goal**: Is the goal of the text clear?
  - **Sections**: Does every section serve its purpose?
  - **Structure**: Is there a coherent narrative?
  - **Argumentation**: Can you follow the points of the report?
  - **Language**: How can the writing quality be improved?
  - **Research Idea**: Comment on what you think about the idea.
  - **Simple Mistakes**: Point out simple improvements such as typos.
  - **Strong Points**: Name at least two good things about the report.
- **Tone**: Be constructive and friendly. Your job is to help your peer to improve their report.
- **Length**: Stay within 150-300 words. Be concise!
- **Format**: Submit as PDF.
- **Effort**: Should take you about two hours per review.

Report Proposal Requirements

Two weeks after you receive your topic you have to submit a report proposal.

- **Content**: Show your plan for the final report:
  - **Topic**: Summarize the topic concisely. What method / system / solution do you present?
  - **Motivation**: Show the relevance of the problem and where it occurs.
  - **Main Part**: What will you focus on in your report?
  - **Goal**: What will the goal of your final report be?
  - **Related Work**: Cite relevant related literature.
- **Length**: Stay within 150-400 words. References do not count.
- **Format**: The same as the final report.

Show that you understood your topic and know how to work on your report.

Presentation Requirements

You should present key points of your topic:

- **Content**: Key points of your focus topic papers. Teach us what we can learn from this paper. You may limit the presentation scope to your focus topic.
- **Audience**: Optimize your presentation for your peers.
- **Scope**: It might be necessary to limit the content of your presentation. Make sure the audience can learn something new. It is not necessary to include everything.
- **Non-Goal**: Do not try to impress the audience by showing how smart you are or how much you know.
- **Format**: Submit as PDF. You will present using our laptop.
- **Time**: Stay within 13-15 minutes. We will cut you off after 16 min!

Panel Discussion Requirements

At the end of each session we will have a panel discussion where all presenters will come to the front. You will discuss together *and* with the audience.

- **Content**: You are free to discuss what you want. Some ideas:
  - How do different topics relate to each other?
  - Which solution would you choose for which use-cases?
  - Ideas for further improvements?
  - Ideas for further research?
  - Connection to previous weeks.
  - Crazy ideas for new systems.
- **Moderation**: The presenters are responsible for the moderation.
- **Group**: We will consider the discussion as a group work of all presenters.
- **Form**: Discuss as natural as possible. Do not read prepared text. Interact.
- **Time**: About 25 minutes.

Pre-Session Protocol Requirements

Find all peers that present in your session. Meet once before your session and practice your presentations together. Prepare talking points for the panel discussion.

- **Content**:
  - **Topics**: For each topic, how does it relate to your topic? (1-2 bullet points each)
  - **Panel Discussion Points**: Possible points to discuss in the sessions. (3-5 bullet points)
  - **Presentation Practice**: What will you change after presenting in front of your peers? (1-2 bullet points)
  - **Presentation Feedback**: What will you change after hearing feedback of your peers? (1-2 bullet points)
  - **Other Presentations**: What did you observe that can be improved? Do not mention names. (1-2 bullet points)
- **Length**: 7-17 bullet points.
- **Format**: Submit as PDF.

Deadlines

Deadlines for everyone (See Timeline):

- Topic Preferences
- Report Proposal
- Report Draft
- Peer Reviews
- Final Report

Deadlines based on your presentation date:

- Pre-Session Protocol: 2 days before your presentation.
- Presentation Slides: 1 day before your presentation.

All deadlines are at 23:59.

Supervision

- Ask organizational questions in the discussion channel on mattermost.
- Ask short questions about your topic and work as direct message on mattermost.
- Meet your supervisor in person and discuss:
  - Once before the submission of your draft.
  - Once before your presentation.
- Usually personal meetings will improve your final grade significantly.

Session Schedule

The structure of a presentation session:

- 3 Presentations per session for 60 min.
  - Present for 13-15 min. (I will stand up after 15, cut you off after 16 min)
  - Discussion for questions regarding the presentation or topic for 2-4 min.
- Short break for 5 min.
- Panel discussion for 25 min.
  - Everybody is welcome to participate in the discussion.
  - Questions regarding individual presentations are allowed.
  - Presenters are responsible to maintain an interesting discussion.

Next Steps

- Make sure you are in the mattermost team Mattermost Team MDBS26
- Find your preferred session topics and focus papers on gitlab:
  https://gitlab.db.in.tum.de/Rieger/mdbs26
- Rank your top three favorite session topics and focus topics and send them to @rieger
  Your message should look like this:

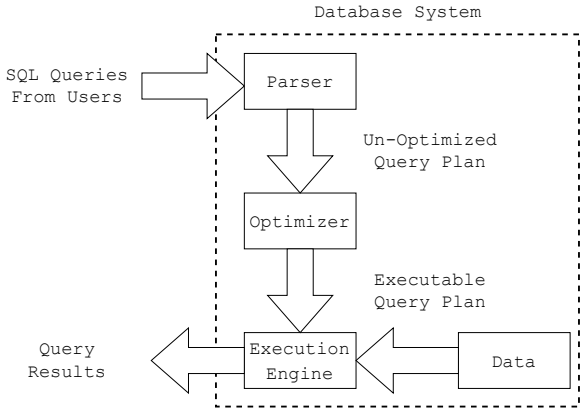
```
My topic preferences:
1: Systems: Neojoin
2: Scheduling: CHARM
3: Optimizers: ORCA
```

- Spend enough time to find a topic you really want to work on!
- Deadline is next week!

Introduction to Cloud Databases

What is a Database System?

- Relational Database Management Systems (RDBMS) store data in tables.
- They perform operations written in SQL



Relational DBs for Different Workloads

- **OLTP**: Online Transaction Processing systems handle frequent updates.
  e.g. banking services
- **OLAP**: Online Analytical Processing systems handle analysis of large datasets.
  e.g. business analytics and recommendations for large online shops
- **HTAP**: Hybrid transactions/analytical processing systems support both workloads.
  e.g. business analytics and recommendations for large online shops on live data

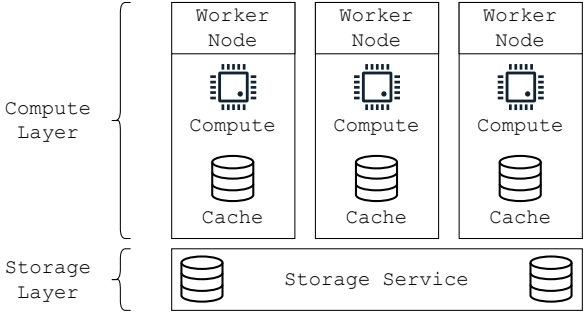
What what makes a good DB?

There are different requirements for DBs:

- Fast answers to queries
- Cheap answers to queries (as in cloud cost)
- Predictable pricing
- Simple setup (serverless)
- Supporting huge datasets

What Changes in a Distributed System?

- Multiple nodes in a cluster
- Network communication



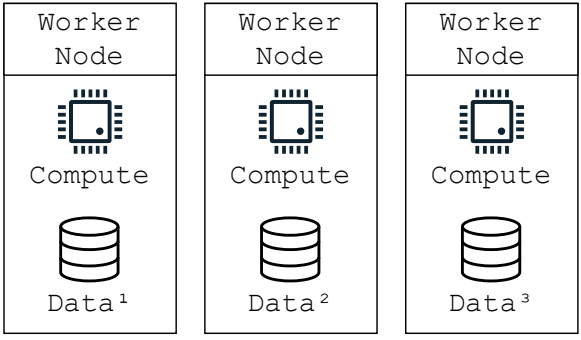
Distributed Architectures

- Shared Nothing
- Decoupled Storage
- Decoupled State and Storage

Distributed Architectures: Shared Nothing

Shared Nothing

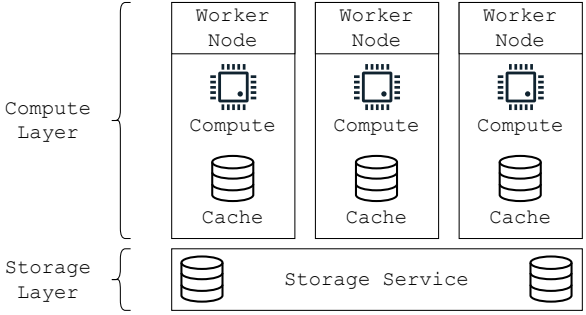
- Data is partitioned to nodes
- Changing cluster size difficult
- If a single node fails the whole system fails



Distributed Architectures: Decoupled Storage

Decoupled Storage

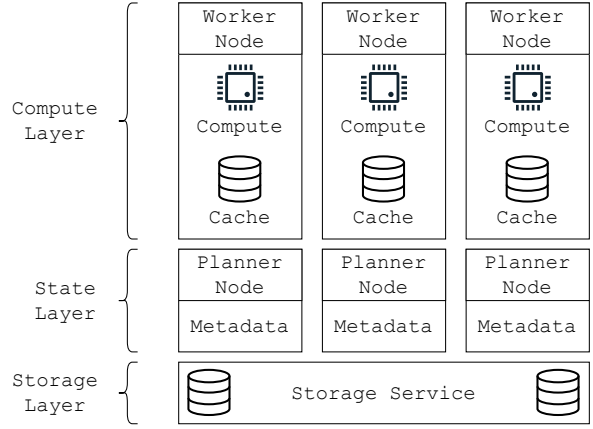
- Data is only cached at nodes
- Changing cluster size easy
- If a single node fails current queries will fail, but the cluster stays intact



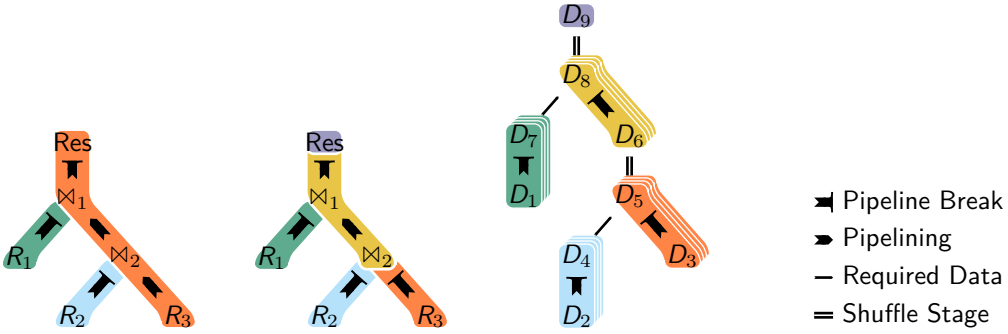
Distributed Architectures: Decoupled State and Storage

Decoupled State and Storage

- Intermediate results are materialized to storage service
- Changing cluster size is easy
- If a single compute node fails even current queries can resume from intermediate state
- May scale to thousands of nodes
- Might have significant overhead
- Microsoft Polaris does this



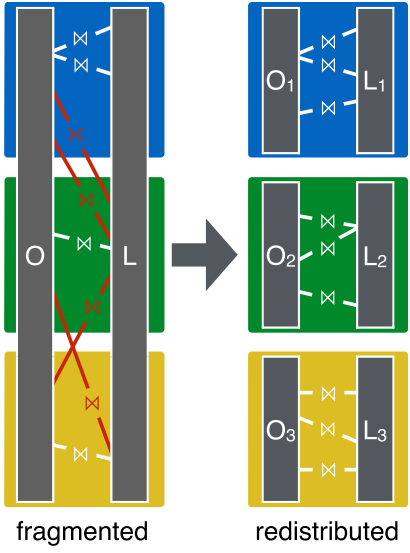
Distributed Query Execution



- Pipelines can be executed in parallel (Scan, Select, Map,...)
- Some operators require data shuffle between nodes (Joins, Aggregations,...)

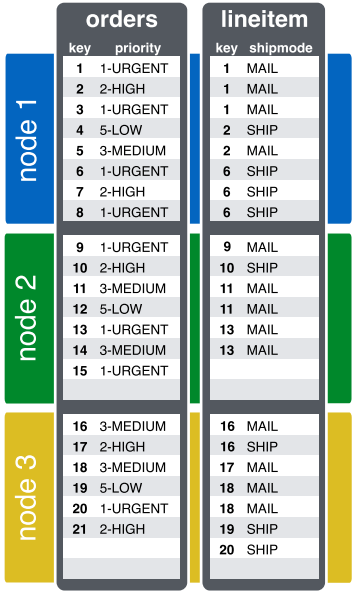
Distributed Joins (1)

- Tuples may need to join with tuples on remote nodes
- Repartition and redistribute both relations on join key
- All potential join-partners will be in same partition



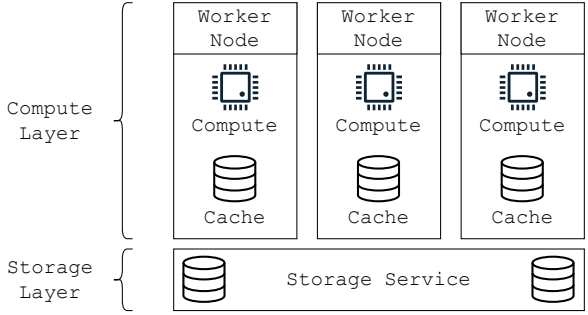
Distributed Joins (2)

- Tuples may need to join with tuples on remote nodes
- Repartition and redistribute both relations on join key
- All potential join-partners will be in same partition



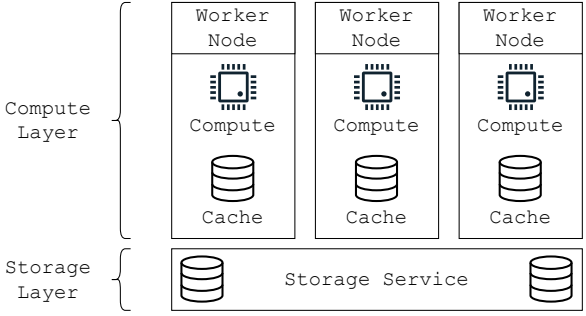
What Changes in the Cloud?

- Storage services allow cheap highly available storage of large datasets (Amazon S3)
- On demand use of compute nodes to save money (Amazon EC2)
- Available cloud instances and services dictate architecture
- Typical x86 cloud instance c5n.18xlarge: 72 cores, 192 GiB RAM, 100 gbit/s network, ≈$3.9/h



Storage Service: What are Object Stores?

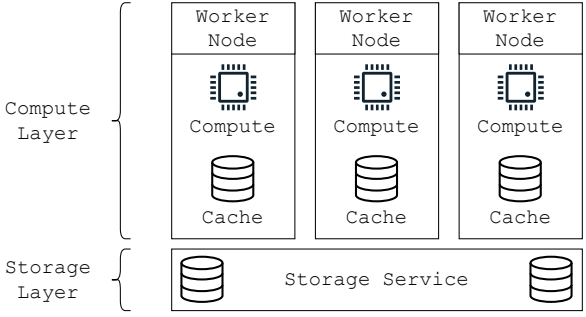
- **Objects** (blobs) are chunks of data (<=5TiB)
  - Read and write objects by their key
  - You can overwrite objects but not modify them
- Implemented as distributed systems
  - Very high availability
  - Very high bandwidth for concurrent requests
- Pricing based on used storage and per request
  - Much cheaper than running storage servers
- There exist many alternatives (Amazon S3, Azure Blobs, Google Cloud Storage, Backblaze B2)



System Properties

Modern systems may have some of the following properties:

- All data on storage service
- Ephemeral compute nodes to adjust to workload
- Heterogeneous nodes (storage, compute, network)
- Node-local cache of data and intermediate results
- Parallel execution of single queries (intra-level parallelism)
- Intermediate result materialization to storage service (recover large queries)



Partitioning Schemes

ID	Name	Age	Country	Occupation
1	John	30	USA	Engineer
2	Alice	25	UK	Teacher
3	Bob	35	Canada	Doctor
4	Maria	28	Brazil	Scientist
5	Emily	40	Australia	Lawyer

Vertical (Projections)

ID	Name	Age
1	John	30
2	Alice	25
3	Bob	35
4	Maria	28
5	Emily	40

ID	Country	Occupation
1	USA	Engineer
2	UK	Teacher
3	Canada	Doctor
4	Brazil	Scientist
5	Australia	Lawyer

- Vertica combines projections with horizontal partitioning.

Horizontal

ID	Name	Age	Country	Occupation
1	John	30	USA	Engineer
2	Alice	25	UK	Teacher

ID	Name	Age	Country	Occupation
3	Bob	35	Canada	Doctor
4	Maria	28	Brazil	Scientist
5	Emily	40	Australia	Lawyer

- May be Hash or Range Partitioning
- Allows distributing data processing
- Used by almost all DBs

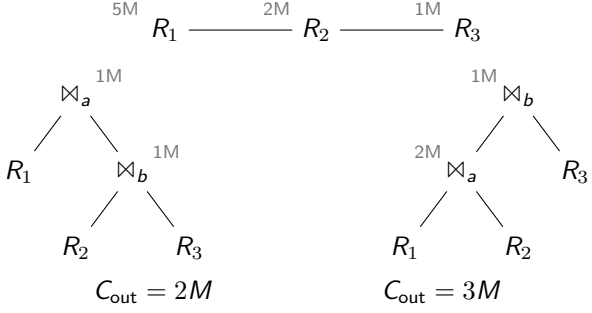
What is a Query Optimizer?

- SQL is declarative
- Many different execution plans compute the correct results
- How do we find the cheapest one?

Example Minimizing Cout

- One of the most important optimizations is join ordering:

```
SELECT *
FROM R1, R2, R3
WHERE R1.a = R2.a
  AND R2.b = R3.b
```



- Size of intermediate results approximates execution time (very roughly!)

Challenges for Optimizers

- Join ordering is an NP-hard problem
- Optimization time must be very quick
  - Queries may contain many joins, but execute very quickly
  - Latency might be dominated by optimization
- Cardinality estimation

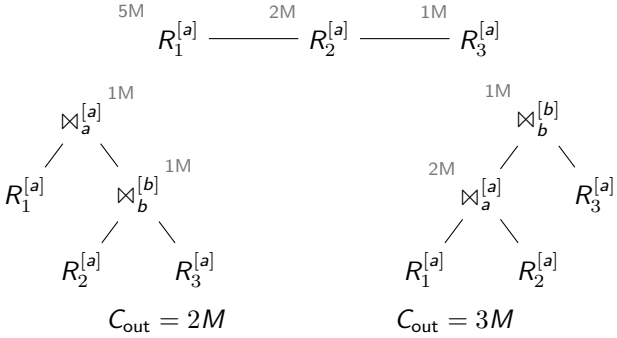
Cardinality Estimation

- We don't know exact sizes of intermediate results
- Systems use statistics of data
- Seems impossible for deep query trees behind join and group-by operators
- Also has to be very fast
- State of the art:
  - Works well about 2-3 joins deep
  - Uses a sample of the data and other statistics

What's New for Distributed Optimizers?

- Cost function
- Network vs. compute
- Search space much larger
- New problem of partition assignment

Example Distributed Cost Function



- Re-partitioning might be very expensive
- Re-partitionings can be avoided with good join order
- C_{out} does not show this cost
- We might want to approximate execution time directly

How to Write a Paper

The Harsh Truth

"When you understand that nobody wants to read your shit, your mind becomes powerfully concentrated. You begin to understand that writing/reading is, above all, a transaction. The reader donates his time and attention, which are supremely valuable commodities. In return, you the writer must give him something worthy of his gift to you."

Steven Pressfield

What You Might Write

- **Thesis**: Bachelor, Seminar, Master, PhD
- **Research**: Research paper, project proposal
- **Industry**: Documentation, design document, white paper, blog post, website

Writing is Thinking

- Writing forces you to make thoughts concrete
- Writing organizes thoughts
- Clear writing is explaining
- Helps you to understand
- Along the process you will have new ideas
- Can even improve your code
- Writing is hard

Good Writing is Important

- More and more papers are written and published
- Attention is scarce
- Effects of bad writing:
  - Reader slows down, re-reads
  - Reader does not understand
  - Reader becomes frustrated
  - Reader stops reading
- Badly written papers get rejected or ignored (or graded poorly)
- Even if the idea is good, no one will care
- You might experience all of this when reading yourself

Why do We Read Papers?

Reading papers is a critical skill:

- Understand latest developments before they are covered by other media like books
- Learn how to write and communicate ideas
- Learn to read critically:
  Ask the right questions, challenge assumptions
- Synthesize new ideas:
  Inspiration comes more often than not when reading the work of others

What to Extract From a Paper?

- **Research question**: Central message
- **Impact**: Motivation, relevance, and possible impact
- **Contributions**: What is new compared to previous work?
  How applicable is approach in general?
- **New info for yourself**: What can you learn from this paper?
  (e.g. good summary of related work)
- **Conclusions**: Takeaways: Can we build upon this work? If so, how? Ideas for future work?

How to Read

You should read a paper in three passes:

1. Get general idea (5-10 minutes)
   - Read abstract, introduction, headings and subheadings, and conclusion
   - You should know what the paper is about now
2. Understand Content (>= 1 hour)
   - Read full paper, ignore details (e.g. proofs)
   - Find key points, take notes
   - Understand figures
   - Mark references for further reading
   - You should understand the key points now
3. Understand in depth (>= 4 hours)
   - Fully understand everything, pay attention to all details
   - Check related work
   - Imagine your own implementation of the solution
   - Question everything
   - Generate new ideas for your own work

How to Read for a Review in This Course

1. Get general idea (5-10 minutes)
   - Read abstract, introduction, headings and subheadings, and conclusion
   - You should know the main goal of the paper now
2. Understand Content (>= 30 min)
   - Read full report
   - Find key points, take notes
   - Understand figures
   - You should understand the key points now
3. Review Report (>= 20 min)
   - Check each of the required points in the text
   - Take notes
   - You should have all information for your review now

Write Your Paper in Drafts

- You should spend most of your time revising
- The first draft should be done very quickly
  - Avoid writers block
  - The first draft is just for you
- Iterate drafts often
  - Requires extreme concentration
  - Maybe do it every morning for an hour
  - Kill your darlings: Rewrite often

Revising is Difficult

- When reading, it is hard to view your own text form the perspective of a reader
- You can try the following:
- **Dead Trees**: Print your paper and read at another physical location than you write
- **Random**: Don't read from start. Jump to random sections and start revising
- **Read Aloud**: Helps catching issues you are otherwise blind to

Paper Structure

- For the paper structure, please refer to the organization slides

Title

- The title of a paper is the first impression people have
- Should be catchy, intriguing, and meaningful
- Will be used to talk about paper
  "Did you read the Polaris paper?" - Good name
  "There was that serverless architecture paper!" - Content is clear
- Naming things is very important: (components, algorithms, methods, products, ...)
  - To talk about things
  - To think about things
  - To remember things
- The title should give the reader an idea what your paper is about

Abstract

- Readers will only read title and (maybe) abstract to decide whether to read your paper
- Abstract should summarize the whole paper
  - What is it about
  - Motivation
  - Core ideas, methods, and solutions
  - Impact, conclusions, and findings
- Should be as short and expressive as possible

Goal Oriented Writing

- Your text has to **provide value** to the reader
  Why bother writing it otherwise?
- You cannot express everything you have to say in one text
- **Define one clear goal** that your text has to achieve (e.g. answer a research question)
- **Kill your darlings**: Remove text that does not help to achieve the goal
  See Chekhov's Gun: (If the gun is there it will be important)
- **Know your audience**: The text should be optimized for the reader

Structure Top Down

- It's natural to start writing small pieces of text
  - You need to figure out how to combine them
  - Makes it hard to get a good overall structure
- Plan your paper top down
  - Start with a very rough outline
  - Iteratively specify section contents with increasing detail
  - Works for whole paper as well as individual sections

Write Sections with the Onion Principle

- Sometimes you need to explain several independent things to lead to the next point
- Use the onion principle:
  - Start with the main point / complete overview but do not go into detail
  - Add several layers: Each layer may go down further into detail
- This holds for sections as well as the whole paper:
  - Title
  - Abstract
  - Introduction
  - Main part (may contain several layers)

Structure Paragraphs

- The first sentence in a paragraph should introduce its topic
- The rest should discuss it
- Make clear what the point of the paragraph is:
  - After the introduction
  - At the end of the paragraph
- Sometimes you will have to deviate from this structure

Avoid the Wall of Text

- Reader looses attention when reading (or even seeing) long text
- Use subsections
- Use small titles
- Use short paragraphs
- Use bullet points

Examples are Crucial

- Help reader to check whether they understood something
- Should be minimal but interesting
- May be used in whole paper, also as motivation

The Baseline Idea

- When writing about an approach it may be useful to compare against a well-known baseline
- Problems of baseline lead to solution
- This might help with the whole story of a paper

Repeat Important Points

- Readers may read superficially or only parts of the text
- Important points should be clear to them too
- Repeat them in: (abstract), introduction, main part, conclusion

What Makes Text Good?

- Good Writing is clear, easy-to-understand writing
- Minimize the effort and time the reader has to spend to read
- Maximize the value the reader has to gain

Why is Good Writing so Difficult?

- Text is linear - Humans think associatively (like a graph)
- Curse of knowledge:
  - Once you understand something, it is hard to empathize with someone who doesn't

Example: Clear Sentences

- Our lack of knowledge about local conditions precluded determination of committee action effectiveness in fund allocation to those areas in greatest need of assistance. ❌

Example: Clear Sentences

- Our lack of knowledge about local conditions precluded determination of committee action effectiveness in fund allocation to those areas in greatest need of assistance. ❌
- Because we knew nothing about local conditions, we could not determine how effectively the committee had allocated funds to areas that most needed assistance. ✅

Example: Clear Sentences

- Our lack of knowledge about local conditions precluded determination of committee action effectiveness in fund allocation to those areas in greatest need of assistance. ❌
- Because we knew nothing about local conditions, we could not determine how effectively the committee had allocated funds to areas that most needed assistance. ✅
- Why is the second version easier to understand?

Example: Clear Sentences

- Our lack of knowledge about local conditions precluded determination of committee action effectiveness in fund allocation to those areas in greatest need of assistance. ❌
- Because we knew nothing about local conditions, we could not determine how effectively the committee had allocated funds to areas that most needed assistance. ✅
- Why is the second version easier to understand?
- Unclear who does what!

Clear Sentences Like Stories

- People think in stories
- **Who does what**?
  - Clear actors (subjects)
  - Clear actions (verbs)

Actors - Subjects

- Actors should be clear: "Knuth developed TeX" ✅
- Things or representations can be actors:
  - "The compiler tells you..." ✅,
  - "The community observes..." ✅
- Avoid hidden actors:
  - "In your paper there is an explanation for ..." ❌

Actors - Subjects

- Actors should be clear: "Knuth developed TeX" ✅
- Things or representations can be actors:
  - "The compiler tells you..." ✅,
  - "The community observes..." ✅
- Avoid hidden actors:
  - "In your paper there is an explanation for ..." ❌
  - "You explain ... in your paper." ✅

Verbs and Adjectives

- Avoid nouns that are hidden verbs:
  - "We made an analysis of the behavior of ..." ❌

Verbs and Adjectives

- Avoid nouns that are hidden verbs:
  - "We made an analysis of the behavior of ..." ❌
  - "We analyzed the behavior of ..." ✅
- Avoid nouns that are hidden adjectives:
  - "The implementation of this data structure poses difficulties" ❌

Verbs and Adjectives

- Avoid nouns that are hidden verbs:
  - "We made an analysis of the behavior of ..." ❌
  - "We analyzed the behavior of ..." ✅
- Avoid nouns that are hidden adjectives:
  - "The implementation of this data structure poses difficulties" ❌
  - "The implementation of this data structure is difficult" ✅

Not All Nouns are Bad

Use nouns to:

- Refer to the previous sentence:
  - "These *arguments* are convincing" ✅
- Create an object instead of a complicated construct:
  - "I do not understand either his *meaning* or her *intention*" ✅
  - "I do not understand either what he means or what she intends" ❌
- Identify the correct actor (subject):
  - "The fact that I set the correct flags was crucial" ❌
  - "My correct *choice* of flags was crucial" ✅
- Name repeated concepts (create new actors):
  - "*Anyblob* is a download manager" ✅

More Useful Methods

- Be careful with passive voice
- Write specific and concrete
- Fewer prepositions
- Shorter sentences
- Logical order
- Clear logical relationships

Connected Information Flow

- Sentences often start with a transition or evaluation:
  - Hence, but, fortunately, importantly
- The beginning of a sentence should contain known easy to recognize things
- The end of a sentence should contain newest and significant information
- You automatically stress the end of a sentence

Balance Clarity and Flow

- Sometimes clarity and flow are in conflict

1. "The query optimizer chooses a hash join whenever both inputs are large and unsorted."
2. "A hash join is chosen whenever both inputs are large and unsorted."

Balance Clarity and Flow

- Sometimes clarity and flow are in conflict

1. "The query optimizer chooses a hash join whenever both inputs are large and unsorted."
2. "A hash join is chosen whenever both inputs are large and unsorted."

- **(1)** is more clear

1. Relational systems offer several physical operators for evaluating a join.
2. **(1) or (2)**

Balance Clarity and Flow

- Sometimes clarity and flow are in conflict

1. "The query optimizer chooses a hash join whenever both inputs are large and unsorted."
2. "A hash join is chosen whenever both inputs are large and unsorted."

- **(1)** is more clear

1. Relational systems offer several physical operators for evaluating a join.
2. **(1) or (2)**

- **(2)** connects better to the previous sentence: "join"

Be Concise

- Use as few words as possible to express what you mean
  - "In my personal opinion, we must listen to and think over in a punctilious manner each and every suggestion that is offered to us." ❌
  - "We must consider each suggestion carefully." ✅
- Another example:
  - "Imagine a picture of someone engaged in the activity of trying to learn the rules for playing the game of chess." ❌
  - "Imagine someone trying to learn the rules of chess." ✅

Avoid Wordy Phrases

- Avoid wordy phrases:

❌	✅
the reason for, due to the fact that, this is	because, since, why
despite the fact that, regardless of the fact that	although, even though
in the event that	if
on the occasion of	when
it is crucial that	must, should
is able to	can
it is possible that	may, might, can, could
prior to	before
does not have	lacks

Figures Matter

- Figures are an integral part of communication
- Many people only look at figures and captions
- Visuals make papers recognizable: Add a first page figure
- Figures are equally important as text
- Text should discuss the takeaways of a figure

Amazon Redshift and the Case for Simpler Data Warehouses

Anurag Gupta, Deepak Agarwal, Derek Tan, Jakub Kulesza, Rahul Pathak, Stefano Stefani, Vidhya Srinivasan

Amazon Web Services

Abstract

Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse solution that makes it simple and cost-effective to efficiently analyze large volumes of data using existing business intelligence tools. Since launching in February 2013, it has been Amazon Web Service's (AWS) fastest growing service, with many thousands of customers and many petabytes of data under management.

Amazon Redshift's pace of adoption has been a surprise to many participants in the data warehousing community. While Amazon Redshift was priced disruptively at launch, available for as little as $1000/TB/year, there are many open-source data warehousing technologies and many commercial data warehousing engines that provide free editions for development or under some usage limit. While Amazon Redshift provides a modern MPP, columnar, scale-out architecture, so too do many other data warehousing engines. And, while Amazon Redshift is available in the AWS cloud, one can build data warehouses using EC2 instances and the database engine of one's choice with either local or network-attached storage.

In this paper, we discuss an oft-overlooked differentiating characteristic of Amazon Redshift – simplicity. Our goal with Amazon Redshift was not to compete with other data warehousing engines, but to compete with non-consumption. We believe the vast majority of data is collected but not analyzed. We believe, while most database vendors target larger enterprises, there is little correlation in today's economy between data set size and company size. And, we believe the models used to procure and consume analytics technology need to support experimentation and evaluation. Amazon Redshift was designed to bring data warehousing to a mass market by making it easy to buy, easy to tune and easy to manage while also being fast and cost-effective.

1. Introduction

Many companies augment their transaction-processing database systems with data warehouses for reporting and analysis. Analysts estimate the data warehouse market segment at 1/3 of the overall relational database market segment ($14B vs. $40B for software licenses and support), with an 8-11% compound annual growth rate (CAGR)[1]. While this is a strong growth rate for a large, mature market, over the past ten years, analysts also estimate data storage at a typical enterprise growing at 30-40% CAGR. Over the past 12-18 months, new market research has begun to show an increase to 50-60%, with data doubling in size every 20 months.

Figure 1: Data Analysis Gap in the Enterprise [10]

This implies most data in an enterprise is "dark data"[2]: data that is collected but not easily analyzed. We see this as unfortunate. If our customers didn't see this data as having value, they would not retain it. Many companies are trying to become increasingly data-driven. And yet, not only is most data already dark, the overall data landscape is only getting darker. Storing this data in NoSQL stores and/or Hadoop in one way to bridge the gap for certain use cases. However it doesn't address all scenarios.

In our discussions with customers, we found the "analysis gap" between data being collected and data available for analysis as due to four major causes.

1. Cost – Most commercial database solutions capable of analyzing data at scale require significant up-front expense. This is hard to justify for large datasets with unclear value.
2. Complexity – Database provisioning, maintenance, backup, and tuning are complex tasks requiring specialized skills. They require IT involvement and cannot easily be performed by line of business data scientists or analysts.
3. Performance – It is difficult to grow a data warehouse without negatively impacting query performance. Once built, IT teams sometimes discourage augmenting data or adding queries as a way of protecting current reporting SLAs.
4. Rigidity – Most databases work best on highly structured relational data. But a large and increasing percentage of data consists of machine-generated logs that mutate over time, audio and video, not readily accessible to relational analysis.

We see each of the above issues only increasing with data set size. To take one large-scale customer example, the Amazon Retail team collects about 5 billion web log records daily (2TB/day,

[1] Forecast data combined from IDC, Gartner, and 451 Research.
[2] http://www.gartner.com/it-glossary/dark-data

First page figure ✅

Figures Should be Stand-Alone

- It should be possible to understand a figure without the text
- Use captions to explain figures
- Captions may be several lines long
- Use text within figures to explain
- Nonetheless, each figure should be referenced by your text!

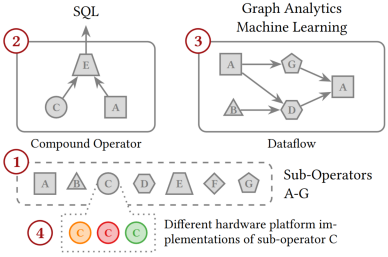


Figure 1: Sub-operators ① build more complex data operations ② or dataflows ③, where each sub-operator can be implemented on multiple hardware platforms ④.

Stand-alone figure ✅

Figures Should be Simple

- Make figures as simple as possible
- Better use two figures than one complex figure

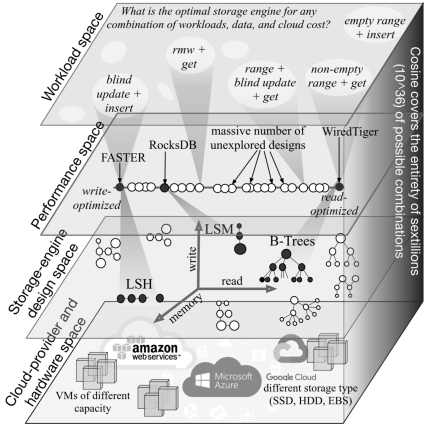
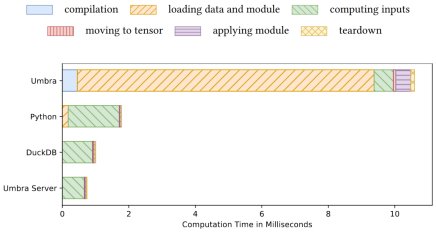


Figure 1: Fixed-design systems capture only a small fraction of the possible storage-engine design space on the cloud.

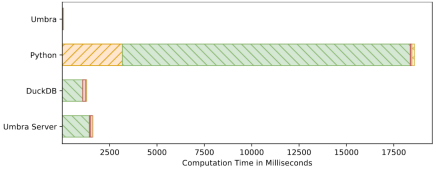
Complex figure ❌

Rely on Color?

- Good coloring can make a graph more understandable
- Should be accessible to colorblind and people with black and white printers
  - Important to many, ignored by many
- Our advice:
  - Use color to make figures as expressive as possible
  - Use shades ($<= 4$), shapes, or text to present the same information without color
  - Ignore this rule if it is impossible



(a) Linear regression execution for 100 tuples



(b) Linear regression execution for 1 million tuples

Figure 10: Runtime decomposition of linear regression.

Colored figure that works on black and white ✅

Revise Figures

- Just as for text, you should spend most of your time revising a figure
- Start with *killer plots* to explore data
- Drill down to the plots that are really useful
- Graphs are comparisons: Make sure it compares what you want to show

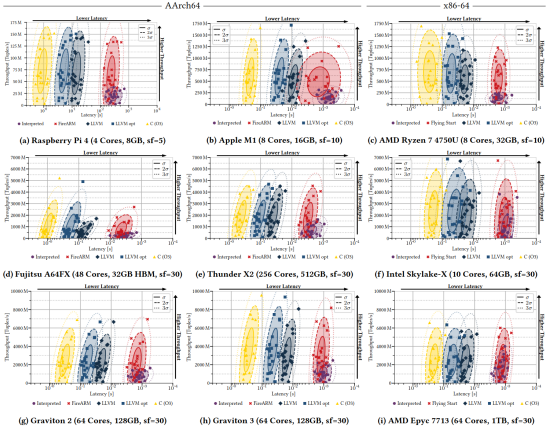


Figure 6: Compile-time and throughput of different query-compilation strategies in Umbra running the TPC-H benchmark.

Killer plot that should be used for exploring data ❌

Graphs and Tables

- Usually, graphs are better than tables
- Tables allow to perform lots of comparisons
  - You can provide a lot of detail in tables
- Use visual hints in tables:
  - Bold for **best**
  - Color for good/neutral/bad
- `\usepackage{booktabs}` works well for tables

Official ScanNet Benchmark

Method	avg IoU	Chair	Floor	Other Furniture	Picture	Sofa	Table	Wall	...
PointNet++	0.339	0.360	0.677	0.183	0.117	0.346	0.232	0.523	...
PointNet++ [1]	0.481	0.686	0.931	0.299	0.102	0.580	0.470	0.711	...
Additional Features (ours)	**0.557**	**0.744**	**0.946**	**0.376**	**0.205**	**0.643**	**0.497**	**0.756**	...

Table with visual hints ✅

Pseudocode

- Pseudocode can be very helpful
- Describes algorithms very accurately
  - Be precise on details, there is no room for interpretation
- Is difficult to understand
  - Reduce cognitive load as much as possible
  - De-clutter syntax (Python-like syntax can be good)
  - Use syntax highlighting
  - Name variables well
  - Name algorithms / functions / methods
  - Add descriptive captions
  - Specify input and output

Algorithm 4: Refining binary join trees

input : An optimized operator tree T
output : A semantically equivalent operator tree T' which may employ multi-way joins

```
1 function refineSubtree(T)
2     if T ≠ T_l ⋈ T_r then
3         return T ;
4     T'_l ← refineSubtree(T_l) ;
5     T'_r ← refineSubtree(T_r) ;
      // Detect growing joins and multi-way join inputs
6     if |T| > max(|T'_l|, |T'_r|) ∨ T'_l ≠ T_l ∨ T'_r ≠ T_r then
7         return collapseMultiwayJoin(T'_l ⋈ T'_r) ;
8     return T'_l ⋈ T'_r ;
```

Pseudocode ✅

Large Language Models (LLMs)

- LLMs can write text
- For now (2025) they lack understanding of *new* things
- They are useful for (re-)formulating your own text
- They are not useful for structuring or creating a convincing story
- You can ask them for 10 variants of a sentence for inspiration
- If you don't want to give some company all your data, you can also self-host small models
  I like `ollama` with `llama` models
- If we notice that your paper includes any text you did not understand or find long sections that just paraphrase sources you will not pass this course

LaTeX

- LaTeX composes visually nice results (visual appeal is important)
- Often, you have to use templates anyway
- Write one sentence per line
  - Check length of sentences
  - Easier reasoning about sentences (each sentence should have a purpose)
  - Works well with version control

Git

- **Always use git** even if you are working alone
- Helps with collaboration
- Synchronizes across devices
- Shows diffs
- Makes it harder to loose data
- Commit often, push often
- Check in all stuff necessary for building (text and figures) but not outputs

Building

- We provide you a template with a makefile
- Error messages are terrible, compile often
- Overleaf can be good, but we recommend building locally
- Typst can be nice, but there is no suitable template

LaTeX is a Time Sink

- Prototyping of any construct that is not text (tables, diagrams) should be done outside of LaTeX
- Sketching on real paper is generally the fastest

Citations with BibTeX

- Citing with BibTeX is easy
- Get correct `.bib` files from dblp.org
  - Be sure that you cite the correct version of the paper
  - Arxiv is one of the worst sources, try to find a source from a journal or conference
  - Especially on google scholar you will often find bad BibTeX files
- Add new bib entries as soon as you `\cite` them (or earlier)

sample.bib

```
@Article{Abril07,
  author        = "Patricia S. Abril and...",
  title         = "The patent holder's...",
  journal       = "Communications of the ACM",
  volume        = "50",
  number        = "1",
  month         = jan,
  year          = "2007",
  pages         = "36--44",
  doi           = "10.1145/1188913.1188915",
  url           = "http://doi.acm.org/...",
}
```

main.tex

```
\section{Citations}
Some examples of references.
A paginated journal article~\cite{Abril07}, ...
```

Final Touches

- New terms should be written once in *italics* and then explained
- Make sure each figure is referenced in text
- Position figures when you are finished with the rest
- Optimize line breaks
- Check references
- Spell check
- Try grammarly (don't blindly follow everything)

Further References

- Deirdre Nansen McCloskey, **Economical Writing**,Third Edition, 2019
- Joseph Williams, **Style: Toward Clarity and Grace**, Univ. of Chicago Press, 1990
- Justin Zobel, **Writing for Computer Science**, Springer, Third Edition, 2014
- Larry McEnerney, **The Craft of Writing Effectively** [link]
- Lorenz Froihofer, **Tips for scientific writing (for Germans)** [link]
- Lorenz Froihofer, **How to write a computer science paper** [link]

Summary

- Good writing is hard but important
- Each sentence needs to contribute to your goal
- Revise more than you write
- Write clear and concise
- A perfect text is one where you cannot omit anything

Presentations

Goal and Scope

Time is limited

- **Limit your scope**: You will not be able to present everything you know
- **Define one clear goal**: What can you reasonably achieve in 15 minutes?
- **Know your audience**: The goal and presentation should be optimized for your audience
- **Pace yourself**: True understanding takes time

Too Much Content

You probably have too much content if you

- **Rush**: You have problems to finish in time
- **Cause confusion**: Your audience does not understand what you mean
- **Cause information overload**: Your audience does not remember or does not want to listen anymore
- **Cover everything you know**

Start

- **Show your scope**: Tell the audience what they can expect from the talk
- **Motivate**:
  - Why is this topic relevant?
  - What problem do we solve?
  - What is new?
  - Why do you care?
- **Contextualize**: What other things exist, how is this different?
- **Spark interest**: Give your audience a reason to want to listen to you

Main Part (1)

- **Tell one story**: Humans are used listening to stories
  - Try to logically connect parts of your talk
- **Logical order**: Structure your talk so the pieces build onto each other
  - Usually this will be different from the chronological order

Main Part (2)

- **Maintain big picture**: It should be clear (to everyone) how each part relates to the main goal of the talk
- **Deep dive**: Go into the interesting details
  - When it contributes to your goal
  - When time allows it
  - When you can make it understandable
  - Examples are very helpful for this
- **Give anchors**: Try to win attention back
  - Nobody is attentive all the time
  - Part of your audience might fail to follow deep dives
  - It should be possible at several points within your talk to follow again
  - Give summaries of challenging parts
  - Make it obvious when one can follow again

End

- **Repeat key points**: Repeat things that everybody should know by the end
- **Highlight implications**: Point audience when their new knowledge is relevant
- **Indicate future work**: Give an outlook of what might come next

Preparation

- **Practice, Practice, Practice**:
  Really!: You will intuitively notice and avoid many mistakes when you practice
- **Record yourself**: To observe your talk from an outside perspective
- **Get the audience perspective**
  1. Find somebody who does not know the topic well
  2. Give the talk to them
  3. They cannot hallucinate missing parts
  4. Ask them what the talk was about
  5. Ask them to repeat the key points from memory
- **Know place and setup**: A great talk can be ruined if you are not there or your setup does not work

Giving the Talk

- **Show interest**: If you show that you care, the audience might as well
- **Speak freely**: Take advantages of live presentations over text or video
- **Face the audience**: Talk to the audience, not the screen or your computer
- **Engage with the audience**: Ask (non-hard) questions, show of hands, ... (if you feel comfortable)
- **Try to avoid**:
  - **Filler words**: A pause is often better than an "uhm"
  - **Fidgeting**: Grab a pen or presenter to occupy your hands
  - **Pacing**: Try to stand on both feet most of the time
  - **Apologizing**: It only draws negative attention and wastes time
    - "I didn't have time to finish this slide"
    - "I am no expert, but this is what I can say"
    - "My computer crashed so I don't have ..."

Handle Questions

- **Answer** if you can
- **Ask for clarification** if needed
- **Repeat** the question for clarification if you are unsure
- **Think** for a few seconds instead of giving a worse answer quickly
- **Admit** if you don't know the answer
  - You can still acknowledge the question
  - Point out why the question is interesting
  - What would one need to do to answer it?
  - Do you know about something similar?

What About the Lecture Slides

- These slides are a horrible example
- We designed them for reference, not only as supplementary material for a talk
- We don't have the time to prepare 90 min lectures as nicely as 15 min talks

Human Focus

Humans can only focus on a single thing well

- Your slides should support your talk, but should not replace it
- Slides do not have to be useful on their own
- Put as few words as possible

Human Ability to Absorb Information

Humans cannot absorb new information very quickly

- Leave enough time for every piece of information to be absorbed
- Plan 1-2 minutes per slide

Readability

- Check font sizes
- Ideally you should be able to easily read everything from the last row
- Test setup ahead of time

References

- Do not spend much time on a "References" slide
- Ideally put them as a footnote on each slide
- Figure sources should also be footnotes on the same slide

Last Slide

- This might be the most important slide of all because it is visible for a long time at the end
- Use it!
  Do not waste it with "Thank You" or "Questions?"
- Give a summary of your talk
- Add visual clues to help the audience remember key parts, for example small versions of figures

Avoid

You should avoid

- **Outline**: People usually do not understand the content
- **Moving animations**: They distract, slow you down, have no upsides
- **Cognitive overload**: Try to identify slides that take time to grasp and improve with:
  - Highlight important things
  - Reveal parts step by step

Tips to Make Good Slides Quickly

- **Plan quickly** create place-holders that you can throw away again
  - Post-it's
  - Ugly slides first, refine later
- **Use as few slides as possible**: Reason about the purpose of every single slide
- **Number your slides**: For orientation of both you and the audience
- **Use a comfortable tool**: Latex might not be worth your time

Further References

- Patrick Winston, **How to speak** [link]
- Elmar Juergens, **How you can predict if your presentation will suck** [link]
- Simon Peyton Jones, **How to give a great research talk** [link]
- Steve Lee (CLIMB), **An Introduction to Oral Scientific Presentations** [link]
- Markus Puschel, **How To Give Strong Technical Presentations** [link]