SCIENCE
PASSION
TECHNOLOGY

Data Integration and Analysis
07 Data Provenane and Blockchain

Matthias Boehm

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMVIT endowed chair for Data Management

Last update: Nov 20, 2020

Announcements/Org

- **#1 Video Recording**
  - Link in **TeachCenter** & **TUbe** (lectures will be public)
  - Optional attendance (independent of COVID)

- **#2 COVID-19 Restrictions** (HS i5)
  - Corona Traffic Light: **RED**
  - Temporarily webex lectures and recording

- **#3 Exercises/Programming Projects**
  - Assigned projects so far
    - 35x SystemDS projects (**64** students)
    - 9 x exercise projects (**15** students)
  - One more Kickoff call for remaining students
    - Email to m.boehm@tugraz.at, invites this weekend

Agenda

- Backlog: Missing Value Imputation
- Motivation and Terminology
- Data Provenance
- Blockchain Fundamentals

Missing Value Imputation

Basic Missing Value Imputation

- **Missing Value**
  - Application context defines if 0 is missing value or not
  - If differences between 0 and missing values, use NA or NaN?
  - Could be a number outside the domain or symbol as '?'

- **Relationship to Data Cleaning**
  - Missing value is error, need to generate **data repair**
  - Data imputation techniques can be used as **outlier/anomaly detectors**

- **Recap: Reasons**
  - **#1 Heterogeneity of Data Sources**
  - **#2 Human Error**
  - **#3 Measurement/Processing Errors**

→

MCAR: Missing Completely at Random
MAR: Missing at Random
MNAR: Missing Not at Random

Basic Missing Value Imputation

- **Missing Completely at Random**
  - Missing values are randomly distributed across all records (independent from recorded or missing values)

ID	Position	Salary ($)	
1	Manager	**null**	(3500)
2	Secretary	2200	
3	Manager	3600	
4	Technician	**null**	(2400)
5	Technician	2500	
6	Secretary	**null**	(2000)

- **Missing at Random**
  - Missing values are randomly distributed within one or more sub-groups of records
  - Missing values depend on the recorded but not on the missing values, and **can be recovered**
  - E.g., missing low salary, age, weight, etc.

ID	Position	Salary ($)
1	Manager	3500
2	Secretary	2200
3	Manager	3600
4	**Technician**	**null**
5	**Technician**	**null**
6	Secretary	2000

- **Not Missing at Random**
  - Missing data depends on the missing values themselves

ID	Position	Salary ($)
1	Manager	3500
2	Secretary	**null**
3	Manager	3600
4	Technician	**null**
5	Technician	2500
6	Secretary	**null**

<= 2400 missing

[Abdulhakim Ali Qahtan, Ahmed K. Elmagarmid, Raul Castro Fernandez, Mourad Ouzzani, Nan Tang: FAHES: A Robust **Disguised Missing Values** Detector. **KDD 2018**]

Basic Missing Value Imputation, cont.

- **Basic Value Imputation**
  - **General-purpose:** replace by user-specified constant, or drop records, or one-hot encode as separate column
  - **Continuous variables:** replace by mean, median
  - **Categorical variables:** replace by mode (most frequent category)

- **Iterative Algorithms** (chained-equation imputation for MAR)
  - Train ML model on available data to predict missing information
    - Initialize with basic imputation (e.g., mean)
    - One dirty variable at a time
    - Feature k → label, split data into training: observed / scoring: missing
    - Types: categorical → classification, continuous → regression
  - Noise reduction: train models over feature subsets + averaging

[Stef van Buuren, Karin Groothuis-Oudshoorn: mice: Multivariate Imputation by Chained Equations in R, **J. of Stat. Software 2011**]

Basic Missing Value Imputation, cont.

- **MICE example**
  - Initialization: fill in the missing values with column mean (w/ or w/o NAs)
  - Iterations: each column per iteration

V1	V2	V3	V4	V5
1	56	2	2	2
2	23	0	0	0
1	NA	0	0	2
2	24	-1	2	NA
NA	22	1	2	0

V1	V2	V3	V4	V5
1	56	2	2	2
2	23	0	0	0
1	25	0	0	2
2	24	-1	2	0.8
1.2	22	1	2	0

V1	V2	V3	V4	V5
1	56	2	2	2
2	23	0	0	0
1	25	0	0	2
2	24	-1	2	0.8
1.2	22	1	2	0

train(y) train(x)

V1	V2	V3	V4	V5
1	56	2	2	2
2	23	0	0	0
1	25	0	0	2
2	24	-1	2	0.8
?	22	1	2	0

← test(x)

DNN Based MV Imputation

[Felix Bießmann et al: DataWig: Missing Value Imputation for Tables, **J. of ML Research 2019**]

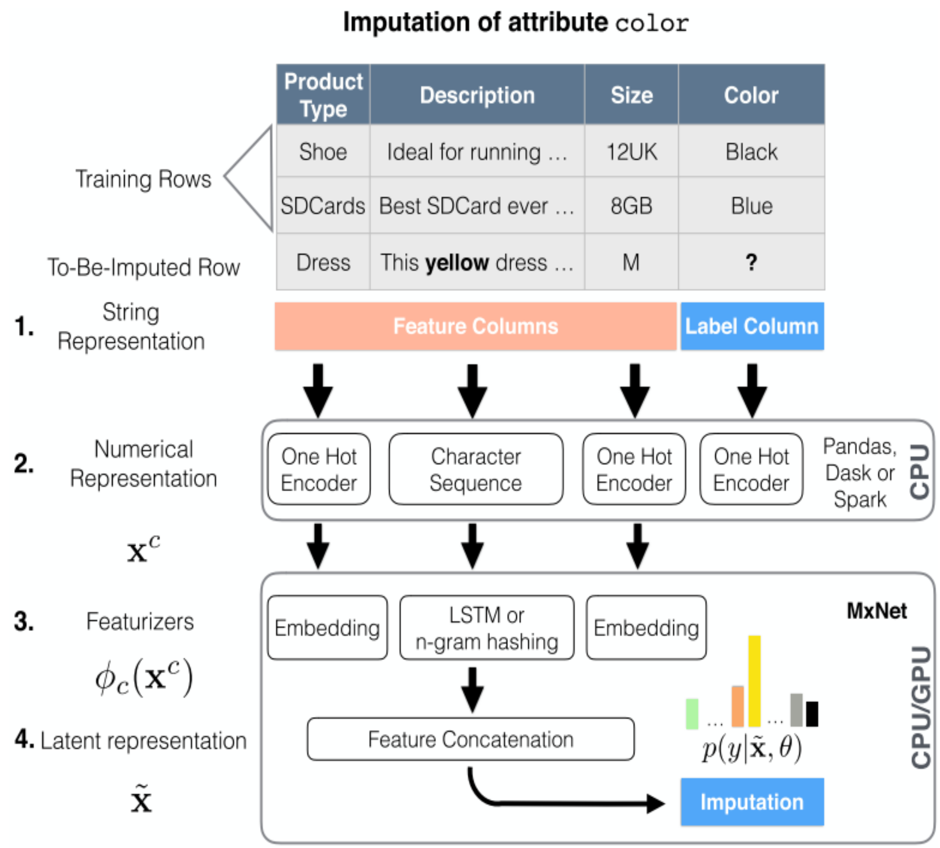
- **DataWig**
  - Missing values imputation for heterogeneous data including unstructured text

Data Type	Featurizers	Loss
Numerical	Normalization Neural Network	Regression
Categorical	Embeddings	Softmax
Text	Bag-of-Words LSTM	N/A

```
table = pandas.read_csv('products.csv')
missing = table[table['color'].isnull()]

# instantiate model and train imputer
model = SimpleImputer(
          input_columns=['description',
                         'product_type',
                         'size'],
          output_columns=['color'])
        .fit(table)

# impute missing values
imputed = model.predict(missing)
```

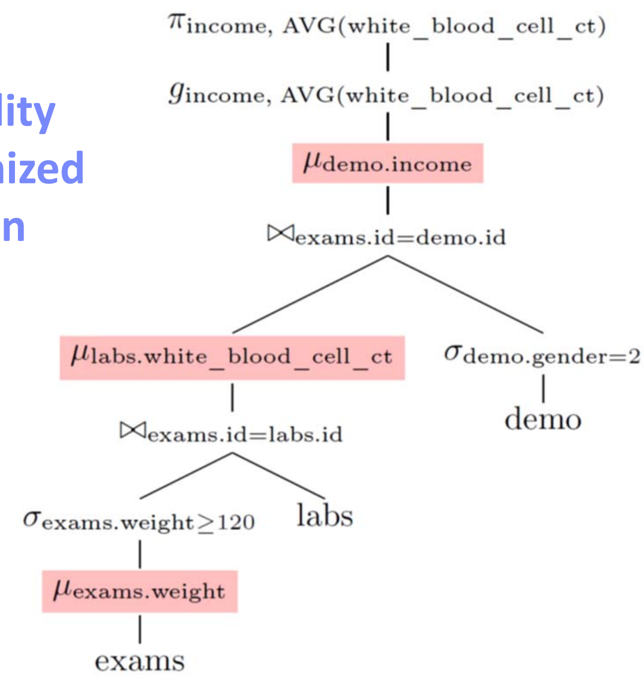


Query Planning w/ MV Imputation

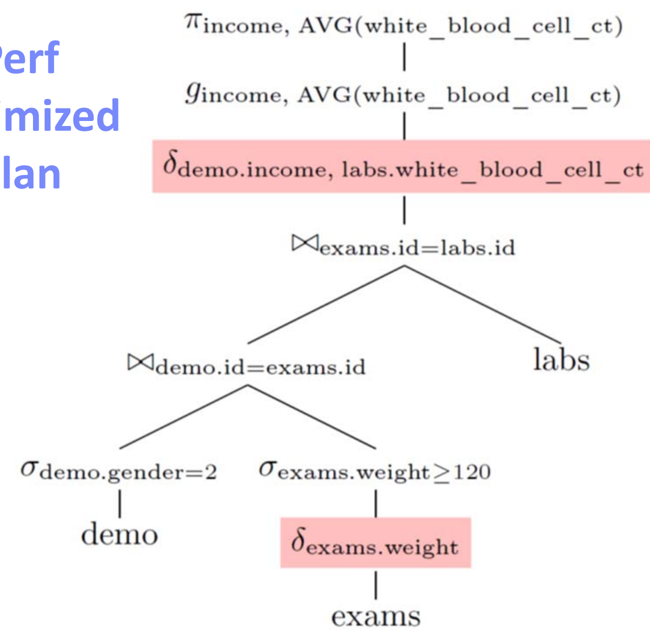
- **Dynamic Imputation**
  - Data exploration w/ on-the-fly imputation
  - Optimal placement of **drop δ** and **impute μ** (**chained-equation imputation** via decision trees)
  - **Multi-objective optimization**

[Jose Cambronero, John K. Feser, Micah Smith, Samuel Madden: Query Optimization for Dynamic Imputation. **PVLDB 2017**]

Quality Optimized Plan



Perf Optimized Plan

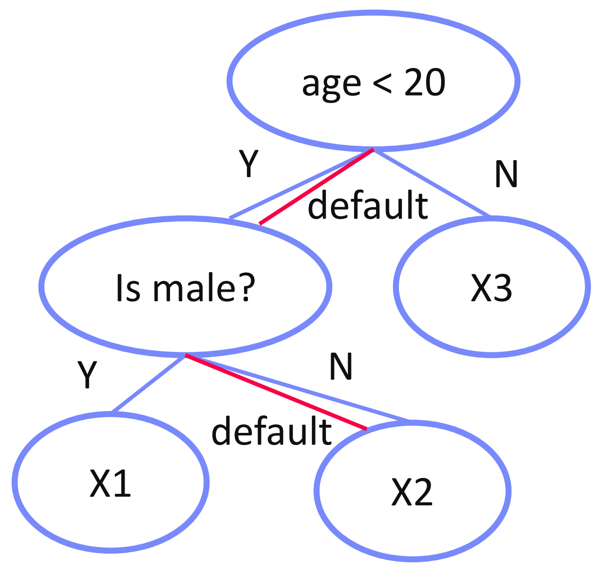


XGBoost’s Sparsity-aware Split Finding

- **Motivation**
  - Missing values
  - Sparsity in general
    (zero values, one-hot encoding)
- **XGBoost**
  - Implementation of gradient
    boosted decision trees
  - Multi-threaded, cache-conscious
- **Sparsity-aware Split Finding**
  - Handles the missing values by
    default paths (learned from data)
  - An example will be classified into the
    default direction when the feature
    needed for the split is missing

[Tianqi Chen and Charlos Guestrin: XGBoost: A Scalable Tree Boosting System, **KDD 2016**]

Example	Age	Gender
X1	?	male
X2	15	?
X3	25	female



Time Series Imputation

[Steffen Moritz and Thomas Bartz-Beielstein: imputeTS: Time Series Missing Value Imputation in R, **The R Journal 2017**]

- **Example R Package imputeTS**

Function	Option	Description
na.interpolation		
	linear	Imputation by Linear Interpolation
	spline	Imputation by Spline Interpolation
	stine	Imputation by Stineman Interpolation
na.kalman		
	StructTS	Imputation by Structural Model & Kalman Smoothing
	auto.arima	Imputation by ARIMA State Space Representation & Kalman Sm.
na.locf		
	locf	Imputation by Last Observation Carried Forward
	nocb	Imputation by Next Observation Carried Backward
na.ma		
	simple	Missing Value Imputation by Simple Moving Average
	linear	Missing Value Imputation by Linear Weighted Moving Average
	exponential	Missing Value Imputation by Exponential Weighted Moving Average
na.mean		
	mean	MissingValue Imputation by Mean Value
	median	Missing Value Imputation by Median Value
	mode	Missing Value Imputation by Mode Value
na.random		Missing Value Imputation by Random Sample
na.replace		Replace Missing Values by a Defined Value

Excursus: Time Series Recovery

- **Motivating Use Case**
  - Given overlapping weekly aggregates y (daily moving average)
  - Reconstruct the **original time series X**

- **Problem Formulation**
  - Aggregates y
  - Original time series X (unknown)
  - Mapping O of subsets of X to y
  - ➔ **Least squares regression problem**

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{O}} \times \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}}_{\mathbf{y}}$$

- **Advanced Method**
  - Discrete Cosine Transform (DCT) (sparsest spectral representation)
  - Non-negativity and smoothness constraints

[Faisal M. Almutairi et al: HomeRun: Scalable Sparse-Spectrum Reconstruction of Aggregated Historical Data. **PVLDB 2018**]

Data Provenance
Motivation and Terminology

Excursus: FAIR Data Principles

[https://www.go-fair.org/fair-principles/]

- **#1 Findable**
  - Metadata and data have globally unique **persistent identifiers**
  - Data describes w/ rich **meta data**; registered/indexes and searchable
- **#2 Accessible**
  - Metadata and data retrievable via open, free and universal **comm protocols**
  - Metadata accessible even when data no longer available
- **#3 Interoperable**
  - Metadata and data use a formal, **accessible, and broadly applicable format**
  - Metadata and data use FAIR vocabularies and qualified references
- **#4 Reusable**
  - Metadata and data described with plurality of accurate and relevant attributes
  - Clear license, **associated with provenance**, meets community standards

Terminology

- **Data Provenance**
  - Track and understand data origins and transformations of data (**where?**, **when?**, **who?**, **why?**, **how?**)

  - Contains meta data, context, and modifications (transform, enrichment)
  - Synonyms: **data lineage**, **data pedigree**

- **Data Catalogs** (curation/**governance**)
  - Directory of datasets including data provenance (meta data, artifacts)
  - Raw/original, curated datasets, derived data products

- **Blockchain**
  - Data structure logging transactions in **verifiable** and **permanent way**

Applications and Goals

a) High-Level Goals

- **#1 Versioning and Reproducibility** (analogy experiments)
- **#2 Explanability, Interpretability, Verification**

b) Low-Level Goals

- **#3 Full and Partial Reuse of Intermediates**
- **#4 Incremental Maintenance of MatViews, Models, etc**
- **#5 Tape/log of Executed Operations** → Auto Differentiation
- **#6 Recomputation for Caching / Fault Tolerance**
- **#7 Debugging via Lineage Query Processing**

Data Provenance

Overview Data Provenance

- **Def Data Provenance**
  - Information about the **origin** and **creation process** of data
- **Example**
  - Debugging suspicious query results

```
SELECT Customer, sum(O.Quantity*P.Price)
  FROM Orders O, Products P
  WHERE O.PID = P.PID
  GROUP BY Customer
```

Customer	Sum
A	7620
B	120
C	130
D	75

OID	Customer	Date	Quantity	PID
1	A	2019-06-22	3	2
2	B	2019-06-22	1	3
3	A	2019-06-22	101	4
4	C	2019-06-23	2	2
5	D	2019-06-23	1	4
6	C	2019-06-23	1	1

PID	Product	Price
1	X	100
2	Y	15
4	Z	75
3	W	120

Overview Data Provenance, cont.

- **An Abstract View**
  - **Data:** schema, structure → data items
  - **Data composition** (granularity): attribute, tuple, relation
  - **Transformation:** consumes inputs, produces outputs
  - **Hierarchical transformations:** query w/ views, query block, operators
  - **Additional:** env context (OS, libraries, env variables, state), users

[Boris Glavic: CS595 Data Provenance – Introduction to Data Provenance, **Illinois Institute of Technology, 2012**]

- **Goal: Tracing of Derived Results**
  - Inputs and parameters
  - Steps involved in creating the result
  - ➔ Store and query data & provenance
  - General Data Protection Regulation (**GDPR**)?

[Zachary G. Ives: Data Provenance: Challenges, Benefits, Research, **NIH Webinar 2016**]


```
1. Read file1
2. Sort rows
3. Compute median
4. Write to file2
```

Prov.

Classification of Data Provenance

- **Overview**
  - Base query $Q(D) = O$ with database $D = \{R_1, \ldots, R_n\}$
  - **Forward lineage query:** $L_f(R_i'', O')$ from subset of input relation to output
  - **Backward lineage query:** $L_b(O', R_i)$ from subset of outputs to base tables
- **#1 Lazy Lineage Query Evaluation**
  - Rewrite (**invert**) lineage queries as relational queries over input relations
  - No runtime overhead but slow lineage query processing
- **#2 Eager Lineage Query Evaluation**
  - Materialize **annotations** (data/transforms) during base query evaluation
  - Runtime overhead but fast lineage query processing
  - Lineage capture: **Logical** (relational) vs **physical** (instrumented physical ops)

[Fotis Psallidas, Eugene Wu: Smoke: Fine-grained Lineage at Interactive Speed. **PVLDB 2018**]

Why-Provenance

[Boris Glavic: CS595 Data Provenance – Provenance Models and Systems, **Illinois Institute of Technology, 2012**]

- **Overview Why**
  - **Goal:** Which input tuples contributed to an output tuple t in query Q
  - Representation: Set of witnesses w for tuple t (**set semantics!**)
    - $w \subseteq I$ (subset of all tuples in instance I)
    - $t \in Q(w)$ (tuple in result of query over w)

- **Example Witnesses**

```
SELECT Customer, Product
  FROM Orders O, Products P
  WHERE O.PID=P.PID
```

	Customer	Date	PID
o1	A	2019-06-22	2
o2	B	2019-06-22	3
o3	A	2019-06-22	2

	PID	Product
p1	1	X
p2	2	Y
p3	4	Z
p4	3	W

	Customer	Product
t1	A	Y
t2	B	W

  - **Witnesses** for **t1**:
    w1 = {o1,p2}, w2 = {o3,p2},
    w3 = {o1,p2,p3}, …, wn = I
  - Minimal witnesses for t1:
    w1 = {o1,p2}, w2 = {o3,p2}

Others: Where/How Provenance

How-Provenance

[Boris Glavic: CS595 Data Provenance – Provenance Models and Systems, **Illinois Institute of Technology, 2012**]

- **Overview**
  - Model how tuples where combined in the computation
  - **Alternative use:** need one of the tuples (e.g., union/projection)
  - **Conjunctive use:** need all tuples together (e.g., join)

- **Provenance Polynomials**
  - **Semiring annotations** to model provenance $(\mathbb{N}[I], +, \times, 0, 1)$

- **Examples**
  - $q = \pi_a(R)$

	a	b
r1	1	2
r2	1	3

→

a
1

r1 + r2

  - $q = \pi_b(R \bowtie S)$

	a	b
r1	1	P
r2	2	G
	3	M

	c	a
s1	S	1
s2	S	2
s3	W	2

→

b
P
G

Provenance Polynomials

r1 x s1

(r2 x s2) + (r2 x s3)

Why Not?-Provenance

[Adriane Chapman, H. V. Jagadish: **Why not? SIGMOD 2009**]

- **Overview**
  - Why are items not in the results
  - **Example Problem:**
    "Window-display-books < $20"
    → (Euripides, Medea).
    → **Why not** (Hrotsvit, Basilius)?

>= 20$?

Author	Title	Price	Publisher
	Epic of Gilgamesh	$150	Hesperus
Euripides	Medea	$16	Free Press
Homer	Iliad	$18	Penguin
Homer	Odyssey	$49	Vintage
Hrotsvit	Basilius	$20	Harper
Longfellow	Wreck of the Hesperus	$89	Penguin
Shakespeare	Coriolanus	$70	Penguin
Sophocles	Antigone	$48	Free Press
Virgil	Aeneid	$92	Vintage

Bug in the query / system?

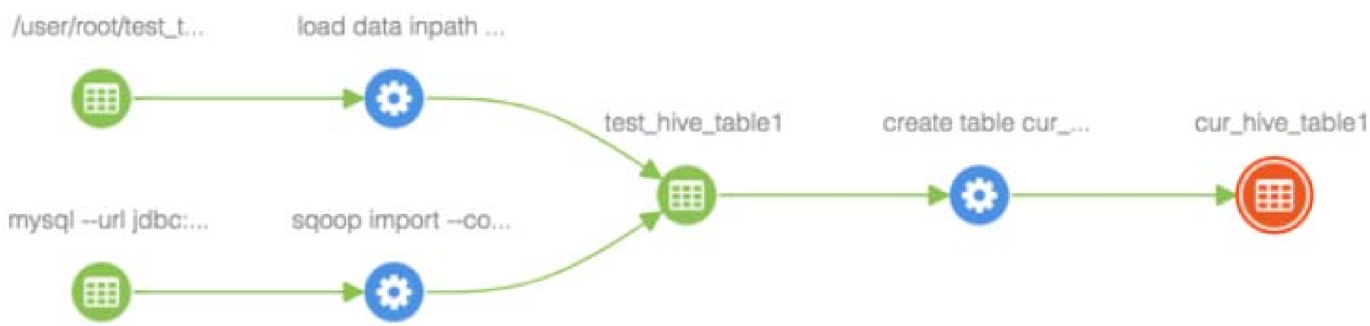
Not in book store?

- **Evaluation Strategies**
  - Given a user question (why no tuple satisfies predicate S), dataset D, result set R, and query Q, leverage **why lineage**
  - **#1 Bottom-Up:** from leafs in topological order to find last op eliminating $d \in S$
  - **#2 Top-Down:** from result top down to find last op, **requires stored lineage**

Apache Atlas

- **Apache Atlas Overview**
  - Metadata management and governance capabilities
  - Build catalog (data classification, cross-component lineage)

- **Example**
  - Configure Atlas hooks w/ Hadoop components
  - Automatic tracking of lineage and side effects



[https://www.cloudera.com/tutorials/cross-component-lineage-with-apache-atlas-across-apache-sqoop-hive-kafka-storm/.html]

Provenance for ML Pipelines (fine-grained)

- **DEX: Dataset Versioning**
  - **Versioning of datasets, stored with delta encoding**
  - Checkout, intersection, union queries over deltas
  - Query optimization for finding efficient plans

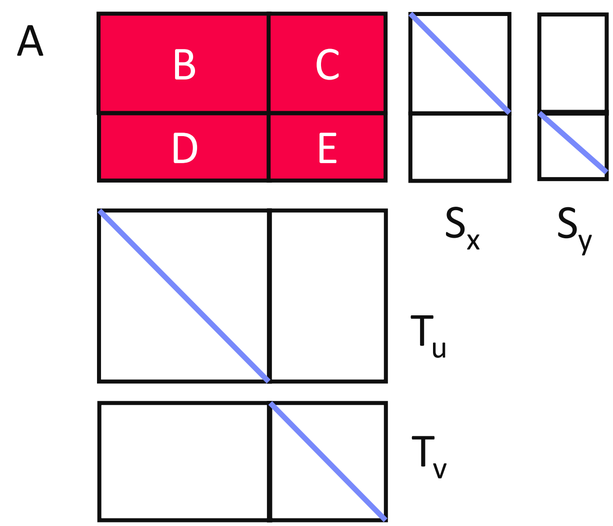
[Amit Chavan, Amol Deshpande: DEX: Query Execution in a Delta-based Storage System. **SIGMOD 2017**]

- **MISTIQUE: Intermediates of ML Pipelines**
  - Capturing, storage, querying of intermediates
  - **Lossy deduplication and compression**
  - Adaptive querying/materialization for finding efficient plans

[Manasi Vartak et al: MISTIQUE: A System to Store and Query Model Intermediates for Model Diagnosis. **SIGMOD 2018**]

- **Linear Algebra Provenance**
  - Provenance propagation by decomposition
  - Annotate parts w/ provenance polynomials (identifiers of contributing inputs + impact)

$$A = S_x B T_u + S_x C T_v + S_y D T_u + S_y E T_v$$



[Zhepeng Yan, Val Tannen, Zachary G. Ives: Fine-grained Provenance for Linear Algebra Operators. **TaPP 2016**]

Provenance for ML Pipelines (coarse-grained)

- **MLflow**
  - Programmatic API for tracking parameters, experiments, and results
  - **autolog()** for specific params

[**Credit:** https://databricks.com/blog/2018/06/05]

```
import mlflow
mlflow.log_param("num_dimensions", 8)
mlflow.log_param("regularization", 0.1)
mlflow.log_metric("accuracy", 0.1)
mlflow.log_artifact("roc.png")
```

- **Flor (on Ground)**
  - DSL embedded in python for managing the workflow development phase of the ML lifecycle
  - DAGs of Actions, Artifacts, and Literals
  - Data context generated by activities in Ground

[Credit: https://rise.cs.berkeley.edu/projects/jarvis/]

[Joseph M. Hellerstein et al: Ground: A Data Context Service. **CIDR 2017**]

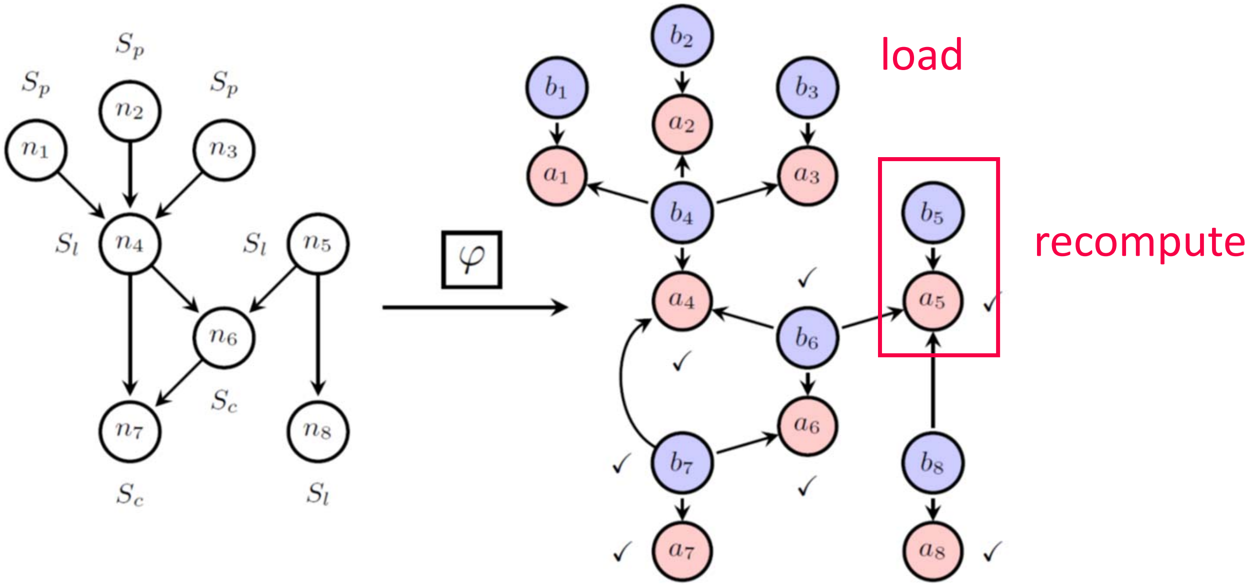
- **Dataset Relationship Management**
  - **Reuse**, **reveal**, **revise**, **retarget**, **reward**
  - Code-to-data relationships (data provenance)
  - Data-to-code relationships (potential transforms)

[Zachary G. Ives, Yi Zhang, Soonbo Han, Nan Zheng,: Dataset Relationship Management. **CIDR 2019**]

Provenance for ML Pipelines (coarse-grained), cont.

- **HELIX**
  - Goal: focus on iterative development w/ small modifications (trial & error)
  - Caching, reuse, and recomputation
  - Reuse as **Max-Flow problem** → **NP-hard** → heuristics
  - Materialization to disk for future reuse

[Doris Xin, Stephen Macke, Litian Ma, Jialin Liu, Shuchen Song, Aditya G. Parameswaran: Helix: Holistic Optimization for Accelerating Iterative Machine Learning. **PVLDB 2018**]



Fine-grained Lineage in SystemDS

- **Problem**
  - **Exploratory data science** (data preprocessing, model configurations)
  - **Reproducibility** and **explainability** of trained models (data, parameters, prep)
  - ➔ **Lineage/Provenance as Key Enabling Technique:** Model versioning, reuse of intermediates, incremental maintenance, auto differentiation, and debugging (query processing over lineage)

- **Efficient Lineage Tracing**
  - Tracing of inputs, literals, and **non-determinism**
  - **Trace lineage of logical operations** for all live variables, store along outputs, program/output reconstruction possible:

```
X = eval(deserialize(serialize(lineage(X))))
```

  - **Proactive deduplication** of lineage traces for loops

Fine-grained Lineage in SystemDS, cont.

- **Full Reuse of Intermediates**
  - Before executing instruction, probe output lineage in cache `Map<Lineage, MatrixBlock>`
  - Cost-based/heuristic caching and eviction decisions (compiler-assisted)

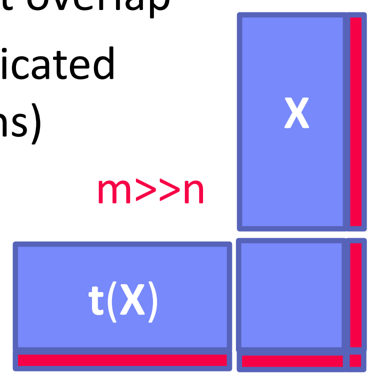
$O(k(mn^2+n^3)) \rightarrow O(mn^2+kn^3)$

```
for( i in 1:numModels )
  R[,i] = lm(X, y, lambda[i,], ...)
```

```
m_lmDS = function(...) {
  l = matrix(reg,ncol(X),1)
  A = t(X) %*% X + diag(l)
  b = t(X) %*% y
  beta = solve(A, b) ...}
```

- **Partial Reuse of Intermediates**
  - **Problem:** Often partial result overlap
  - Reuse partial results via dedicated rewrites (compensation plans)
  - Example: steplm

m>>n



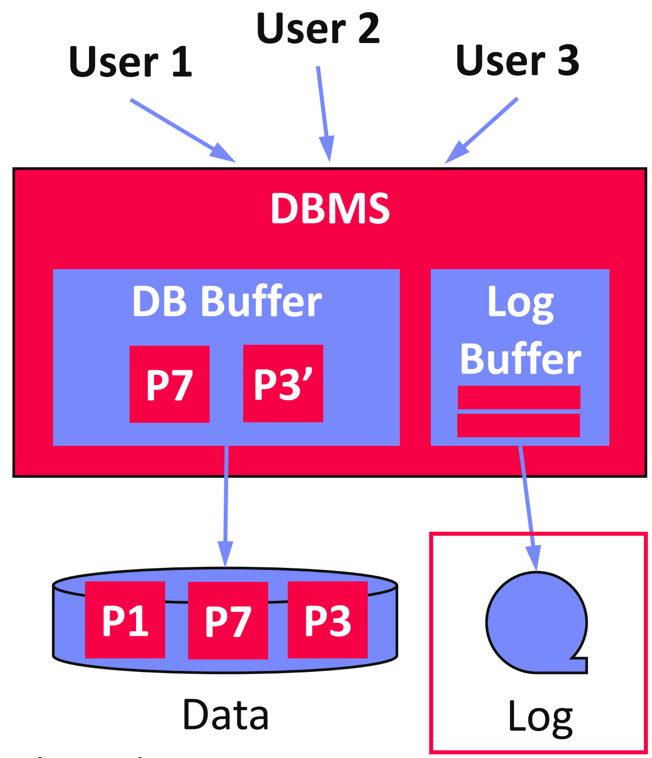

```
m_steplm = function(...) {
  while( continue ) {
    parfor( i in 1:n ) {
      if( !fixed[1,i] ) {
        Xi = cbind(Xg, X[,i])
        B[,i] = lm(Xi, y, ...)
    } }
    # add best to Xg
    # (AIC)
} }
```

$O(n^2(mn^2+n^3)) \rightarrow O(n^2(mn+n^3))$

Blockchain Fundamentals

Recap: Database (Transaction) Log

- **Database Architecture**
  - Page-oriented storage on disk and in memory (DB buffer)
  - Dedicated **eviction algorithms**
  - Modified in-memory pages marked as dirty, flushed by cleaner thread
  - Log: append-only TX changes
  - Data/log often placed on different devices and periodically archived (backup + truncate)



- **Write-Ahead Logging (WAL)**
  - The log records representing changes to some (dirty) data page must be on **stable storage before the data page** (UNDO - atomicity)
  - **Force-log on commit** or full buffer (REDO - durability)
  - **Recovery:** forward (REDO) and backward (UNDO) processing
  - Log sequence number (LSN)

[C. Mohan, Donald J. Haderle, Bruce G. Lindsay, Hamid Pirahesh, Peter M. Schwarz: ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging. **TODS 1992**]

Bitcoin and Blockchain

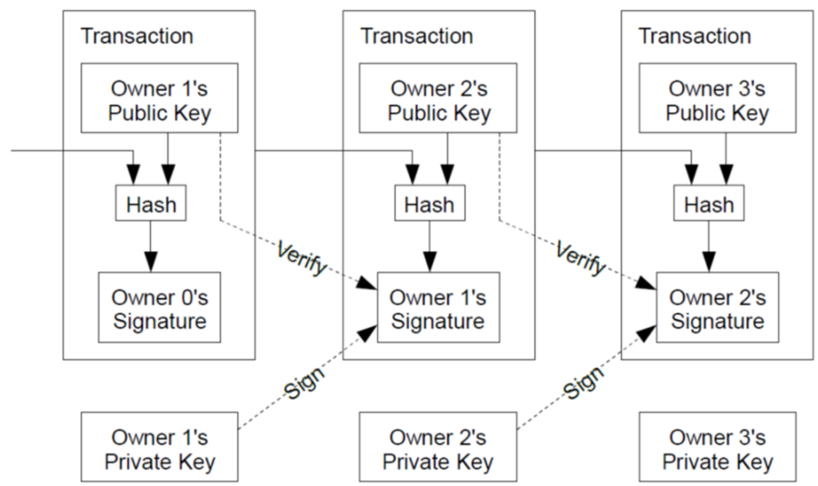
[**Satoshi Nakamoto**: Bitcoin: A Peer-to-Peer Electronic Cash System, **White paper 2008**]

- **Motivation**
  - Peer-to-peer (decentralized, anonymous) electronic cash/payments
  - **Non-reversible transactions** w/o need for trusted third party
  - **Statistics**

	Market Price (USD)	Average Block Size	Transactions per Day	Mempool Size
Nov 21 2019:	$7,862.72 USD	1.16 Megabytes	303,921 Transactions	11,304,890 Bytes
Nov 19 2020: (Paypal Oct 2020)	$17,975.24 USD	1.29 Megabytes	310,424 Transactions	19,920,773 Bytes

[https://www.blockchain.com/charts]

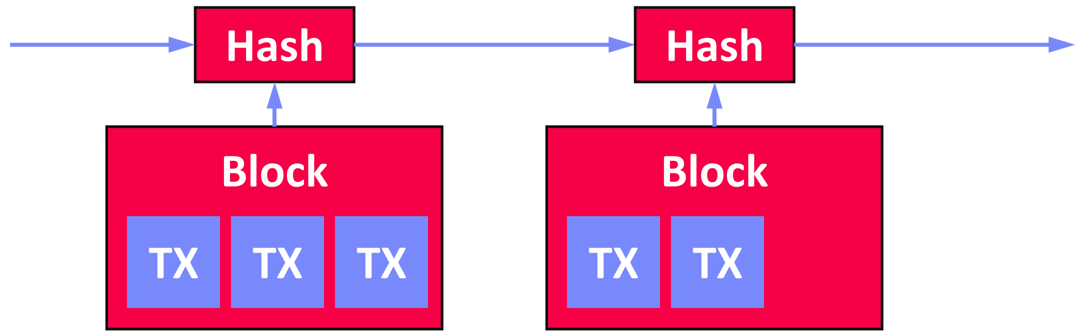
- **Transaction Overview**
  - Electronic coin defined as chain of digital signatures
  - Transfer by signing hash of previous TX and public key of next owner
  - **Double-spending problem** (without global verification)



Blockchain Data Structure

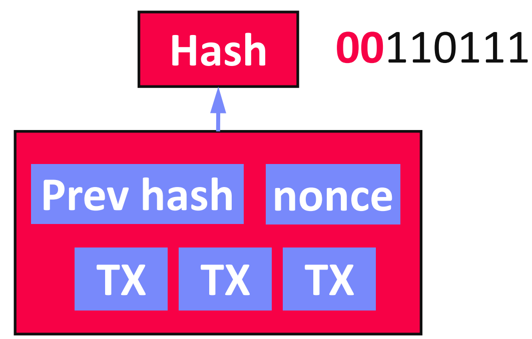
[**Satoshi Nakamoto**: Bitcoin: A Peer-to-Peer Electronic Cash System, **White paper 2008**]

- **Timestamp Server**
  - Decentralized timestamp server: chain of hashes ➔ **public ledger**



Enforces order dependency ➔ No double-spending

- **Proof-of-Work**
  - Scanning for value (nonce) which **SHA-256 hash** begins with a number of zero bits → exponential in number of zeros
  - # zero bits determined by MA of avg blocks/hour
  - Hard to recompute for chain, easy to check
  - **Majority decision:** CPU time, longest chain



Merkel tree (hash tree)

Blockchain Data Structure, cont.

- **Bitcoin Mining**
  - HW: from CPU to GPUs/FPGAs/ASICs (**10-70 TH/s** @ 2-3KW)
  - Usually mining pools → **"mining cartels"**

- **Hash Rate of Bitcoin Network**
  - **~10 min per block** (144 blocks per day)

[https://www.blockchain.com/en/charts/hash-rate?daysAverageString=7&timespan=180days, **Nov 21 2019**]

Blockchain Communication

[**Satoshi Nakamoto**: Bitcoin: A Peer-to-Peer Electronic Cash System, **White paper 2008**]

- **Networking Protocol**
  - New **TXs are broadcast** to all nodes
  - Each node collects new **TXs into a block**
  - Each node works on finding **proof-of-work** for its block
    → **Incentive:** 1st TX in block new coin
    (halves every 210k blocks) for the block creator + TX fees
  - When a node finds a proof-of-work, **broadcast the block** to all nodes
  - Nodes accept the block if **all TXs are valid** (double spending)
  - Nodes express **acceptance by working on next block** in the chain, using the hash of the accepted block as the previous hash

2008: 50BTC
2012: 25BTC
2016: 12.5BTC
2020: 6.25BTC

- **Fault Tolerance**
  - **TX broadcasts:** no need to reach all but many → next block contains it
  - **Block broadcast:** no need to reach all → next block references it

Smart Contracts (Ethereum)

- **Motivation**
  - **Problem:** Bitcoin TXs for transferring X $BTC from A to B (exchange as assets)
  - **Goals:** voting, auctions, games, bets, legal agreements (notary)

- **Ethereum**
  - Decentralized platform that allows creation, management, and execution of smart contracts
  - Ether cryptocurrency, block mining rate: seconds (5 ETH/block)

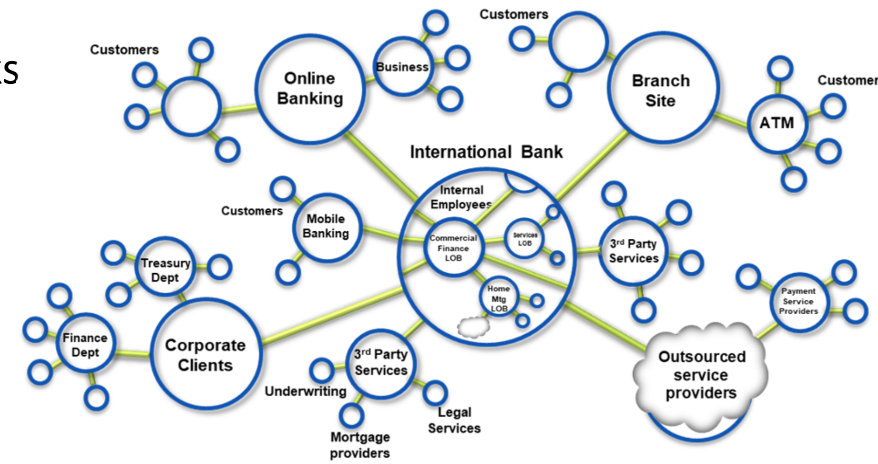
[**Credit:** Shana Hutchison]

- **Smart Contract**
  - Store smart contract (turing-complete programs) on the blockchain
  - **On transfer/trigger:** run smart contract (in) in Ethereum Virtual Machine
  - Language: Serpent/Solidity (deterministic, w/ control flow and function calls)
    ➔ Problem: while(true) → EVM gas and fees (start gas, gas price)
  - Immutability guarantees persistence

Permissioned/Private Blockchains

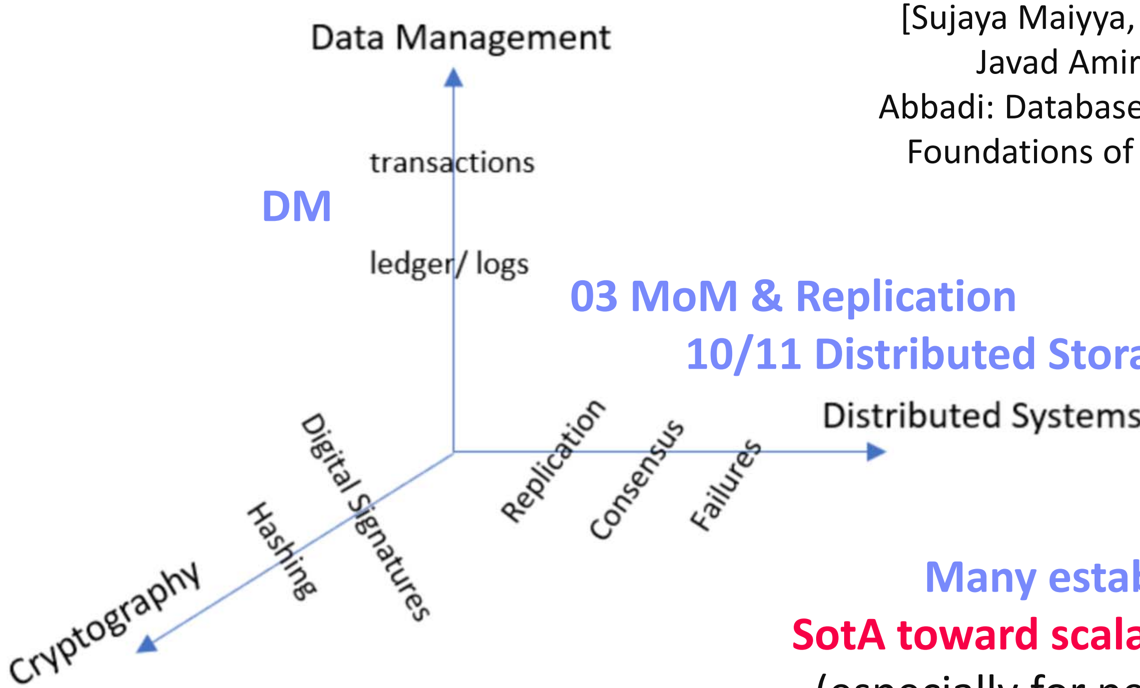
- **Private Setup**
  - Business Networks connect businesses
  - **Participants with Identity**
  - Assets flow over business networks
  - Transactions describe exchange or change of states of assets
  - **Smart contracts** underpin transactions
  - Blockchain as shared, replicated, permissioned ledger (TX log): **consensus**, **provenance**, **immutability**

[**C. Mohan:** State of Permissionless and Permissioned Blockchains: Myths and Reality, **2019**]



- **Hyperledger Fabric (https://github.com/hyperledger/)** HYPERLEDGER
  - IBM, Oracle, Baidu, Amazon, Alibaba, Microsoft, JD, SAP, Huawei, Tencent
  - **Blockchain-as-a-Service** (BaaS) offerings: distributed ledger, libs, tools

Discussion Blockchain



DM

03 MoM & Replication

10/11 Distributed Storage/Compute

[Sujaya Maiyya, Victor Zakhary, Mohammad Javad Amiri, Divyakant Agrawal, Amr El Abbadi: Database and Distributed Computing Foundations of Blockchains. **SIGMOD 2019**]

Many established techniques

SotA toward scalable/efficient blockchains

(especially for permissioned blockchains)

→ **Recommendation: Investigate business requirements/context, decide on technical properties and acceptable trade-offs**

Summary and Q&A

- **Summary**
  - Backlog: Missing Value Imputation
  - Motivation and Terminology
  - Data Provenance
  - Blockchain Fundamentals

- **Next Lectures (Part B)**
  - **08 Cloud Computing Foundations** [Nov 27]
  - **09 Cloud Resource Management and Scheduling** [Dec 04]
  - **10 Distributed Data Storage** [Dec 11]