Data Management (DM)

Preamble

Each area of computer science can be described as "The study of algorithms and data structures to ..." In this case the blank is filled in with "deal with persistent data sets; frequently too large to fit in primary memory."
Since the mid-1970's this has meant an almost exclusive study of relational database systems. Depending on institutional context, students have studied, in varying proportions:

- Data modeling and database design: e.g., E-R Data model, relational model, normalization theory
- Query construction: e.g., relational algebra, SQL
- Query processing: e.g., indices (B+tree, hash), algorithms (e.g., external sorting, select, project, join), query optimization (transformations, index selection)
- DBMS internals: e.g., concurrency/locking, transaction management, buffer management

Today's graduates are expected to possess DBMS user (as opposed to implementor) skills. These primarily include data modeling and query construction; ability to take an unorganized collection of data, organize it using a DBMS, and access/update the collection via queries. Additionally, students need to study:

- The role data plays in an organization. This includes:
  - The Data Life Cycle: Creation-Processing-Review/Reporting-Retention/Retrieval-Destruction.
  - The social/legal aspects of data collections: e.g., scale, data privacy, database privacy (compliance) by design, de-identification, ownership, reliability, database security, and intended and unintended applications.
- Emerging and advanced technologies that are augmenting/replacing traditional relational systems, particularly those used to support (big) data analytics: NoSQL (e.g., JSON, XML, key-value store databases), cloud databases, MapReduce, and dataframes.

We recognize the existing and emerging roles for those involved with data management, which include:

- Product feature engineers: those who use both SQL and NoSQL operational databases.
- Analytical Engineers/Data Engineers: those who write analytical SQL, Python, and Scala code to build data assets for business groups.
- Business Analysts: those who build/manage data most frequently with Excel spreadsheets.
- Data Infrastructure Engineers: those who implement a data management system (e.g., OLTP).
- “Everyone:” those who produce or consume data need to understand the associated social, ethical, and professional issues.

One role that transcends all of the above categories is that of data custodian. Previously, data was seen as a resource to be managed (Information Systems Management) just like other enterprise resources. Today, data is seen in a larger context. Data about customers can now be seen as belonging to (or in some national contexts, as owned by) those customers. There is

now an accepted understanding that the safe and ethical storage, and use, of institutional data is part of being a responsible data custodian.

Furthermore, we acknowledge the tension between a curricular focus on professional preparation versus the study of a knowledge area as a scientific endeavor. This is particularly true with Data Management. For example, proving (or at least knowing) the completeness of Armstrong's Axioms is fundamental in functional dependency theory. However, the vast majority of computer science graduates will never utilize this concept during their professional careers. The same can be said for many other topics in the Data Management canon. Conversely, if our graduates can only normalize data into Boyce-Codd normal form (using an automated tool) and write SQL queries, without understanding the role that indices play in efficient query execution, we have done a disservice.

To this end, the number of CS Core hours is relatively small relative to the KA Core hours. Hopefully, this will allow institutions with differing contexts to customize their curricula appropriately. For some, the efficient storage and access of data is primary and independent of how the data is ultimately used - institutional context with a focus on OLTP implementation. For others, what is "under the hood" is less important than the programmatic access to already designed databases - institutional context with a focus on product feature engineers/data scientists.

Regardless of how an institution manages this tension we wish to give voice to one of the ironies of computer science curricula. Students typically spend the majority of their educational career reading (and writing) data from a file or interactively, while outside of the academy the lion's share of data, by a huge margin, comes from databases accessed programmatically. Perhaps in the not too distant future students will learn programmatic database access early on and then continue this practice as they progress through their curriculum.

Finally, we understand that while Data Management is orthogonal to Cybersecurity and SEP (Society, Ethics, and Professionalism), it is also ground zero for these (and other) knowledge areas. When designing persistent data stores, the question of what should be stored must be examined from both a legal and ethical perspective. Are there privacy concerns? And finally, how well protected is the data?

Changes since CS 2013

- Rename the knowledge unit from Information Management to Data Management. This renaming does not represent any kind of philosophical shift. It is simply an effort to avoid confusion with Information Systems and Information Technology curricula.
- Inclusion of NoSQL approaches and MapReduce as CS Core topics.
- Increased attention SEP and SEC topics in both the CS Core and KA Core areas.

Core Hours

Knowledge Unit	CS Core Hours	KA Core Hours
The Role of Data	2	
Core Database Systems Concepts	2	1

Data Modeling	2	3
Relational Databases	1	3
Query Construction	2	4
Query Processing		4
DBMS Internals		4
NoSQL Systems		2
Data Security & Privacy		
Data Analytics		
Distributed Databases/Cloud Computing		
Semi-structured and Unstructured Databases		
Society, Ethics, and Professionalism		
Total	**9**	**21**

Knowledge Units

DM-Data: The Role of Data and the Data Life Cycle

CS Core:

1. The Data Life Cycle: Creation-Processing-Review/Reporting-Retention/Retrieval-Destruction (See also: SEP-Social-Context, SEP-Ethical-Analysis, SEP-Professional-Ethics, SEP-Privacy, SEP-Security)

Illustrative Learning Outcomes

CS Core:

1. Identify the five stages of the Data Life Cycle

DM-Core: Core Database System Concepts

CS Core:

1. Purpose and advantages of database systems
2. Components of database systems
3. Design of core DBMS functions (e.g., query mechanisms, transaction management, buffer management, access methods)
4. Database architecture, data independence, and data abstraction

5. Use of a declarative query language
6. Transaction mgmt
7. Normalization
8. Approaches for managing large volumes of data (e.g., noSQL database systems, use of MapReduce) (See also: PDC-Algorithms:2)
9. How to support CRUD-only applications
10. Distributed databases/cloud-based systems
11. Structured, semi-structured, and unstructured databases

KA Core:

12. Systems supporting structured and/or stream content

Illustrative Learning Outcomes

CS Core:

1. Identify at least four advantages that using a database system provides.
2. Enumerate the components of a (relational) database system.
3. Follow a query as it is processed by the components of a (relational) database system.
4. Defend the value of data independence.
5. Compose a simple select-project-join query in SQL.
6. Enumerate the four properties of a correct transaction manager.
7. Articulate the advantages for eliminating duplicate repeated data.
8. Outline how MapReduce uses parallelism to process data efficiently.
9. Evaluate the differences between structured and semi/unstructured databases.

DM-Modeling: Data Modeling

CS Core:

1. Data modeling (See also: SE-Requirements (Ask Titus))
2. Relational data model (See also: MSF-Discrete)

KA Core:

3. Conceptual models (e.g., entity-relationship, UML diagrams)
4. Semi-structured data models (expressed using DTD, XML, or JSON Schema, for example)

Non-Core:

5. Spreadsheet models
6. Object-oriented models (See also: FPL-OOP)
    a. GraphQL
7. New features in SQL
8. Specialized Data Modeling topics
    a. Time series data (aggregation, and join)
    b. Graph data (link traversal)
    c. Techniques for avoiding inefficient raw data access (e.g., “avg daily price”): materialized views and special data structures (e.g., Hyperloglog, bitmap)
    d. Geo-Spatial data (e.g., GIS databases) (See also: SPD-Access)

Illustrative Learning Outcomes

CS Core:

1. Articulate the components of the relational data model
2. Model 1:1, 1:n, and n:m relationships using the relational data model

KA Core:

3. Articulate the components of the E-R (or some other non-relational) data model.
4. Model a given environment using a conceptual data model.
5. Model a given environment using the document-based or key-value store-based data model.

DM-Relational: Relational Databases

CS Core:

1. Entity and referential integrity
    a. Candidate key, superkeys
2. Relational database design

KA Core:

3. Mapping conceptual schema to a relational schema
4. Physical database design: file and storage structures (See also OS-Files)
5. Introduction to Functional dependency Theory
6. Normalization Theory
    a. Decomposition of a schema; lossless-join and dependency-preservation properties of a decomposition
    b. Normal forms (BCNF)
    c. Denormalization (for efficiency)

Non-Core:

7. Functional dependency Theory
    a. Closure of a set of attributes
    b. Canonical Cover
8. Normalization Theory
    a. Multi-valued dependency (4NF)
    b. Join dependency (PJNF, 5NF)
    c. Representation theory

Illustrative Learning Outcomes

CS Core:

1. Articulate the defining characteristics behind the relational data model.
2. Comment on the difference between a foreign key and a superkey.
3. Enumerate the different types of integrity constraints.

KA Core:

4. Compose a relational schema from a conceptual schema which contains 1:1, 1:n, and n:m relationships.
5. Map appropriate file structure to relations and indices.
6. Articulate how functional dependency theory generalizes the notion of key.
7. Defend a given decomposition as lossless and or dependency preserving.
8. Detect which normal form a given decomposition yields.

9. Comment on reasons for denormalizing a relation.

DM-Querying: Query Construction

CS Core:

1. SQL Query Formation
    a. Interactive SQL execution
    b. Programmatic execution of an SQL query

KA Core:

2. Relational Algebra
3. SQL
    a. Data definition including integrity and other constraints specification
    b. Update sublanguage

Non-Core:

4. Relational Calculus
5. QBE and 4th-generation environments
6. Different ways to invoke non-procedural queries in conventional languages
7. Introduction to other major query languages (e.g., XPATH, SPARQL)
8. Stored procedures

Illustrative Learning Outcomes

CS Core:

1. Compose SQL queries that incorporate select, project, join, union, intersection, set difference, and set division.
2. Determine when a nested SQL query is correlated or not.
3. Iterate over data retrieved programmatically from a database via an SQL query.

KA Core:

4. Define, in SQL, a relation schema, including all integrity constraints and delete/update triggers.
5. Compose an SQL query to update a tuple in a relation.

DM-Processing: Query Processing

This knowledge area examines the process of how a Relational DBMS executes a query from its SQL origins, to its relational algebraic equivalence, to its optimized transformed equivalence, to a realized execution plan. This leads to the study of database tuning; the creation of specific indices (and denormalization) to allow for optimized query execution.

KA Core:

1. Page structures
2. Index structures
    a. B+ trees (See also: AL-Fundamentals)
    b. Hash indices: static and dynamic (See also: AL-Fundamentals, SEC-Foundations)
    c. Index creation in SQL
3. Algorithms for query operators

    a. External Sorting (See also: AL-Fundamentals)
    b. Selection
    c. Projection; with and without duplicate elimination
    d. Natural Joins: Nested loop, Sort-merge, Hash join
    e. Analysis of algorithm efficiency (See also: AL-Complexity)
4. Query transformations
5. Query optimization
    a. Access paths
    b. Query plan construction
    c. Selectivity estimation
    d. Index-only plans
6. Parallel Query Processing (e.g., parallel scan, parallel join, parallel aggregation) (See also: PDC-Algorithms)
7. Database tuning/performance
    a. Index selection
    b. Impact of indices on query performance (See also: SF-Performance:3, SEP-Sustainability)
    c. Denormalization

Illustrative Learning Outcomes

KA Core:

1. Articulate the purpose and organization of both B+ tree and hash index structures.
2. Compose an SQL query to create an index (any kind).
3. Articulate the steps for the various query operator algorithms: external sorting, projection with duplicate elimination, sort-merge join, hash-join, block nested-loop join.
4. Derive the run-time (in I/O requests) for each of the above algorithms
5. Transform a query in relational algebra to its equivalent appropriate for a left-deep, pipelined execution.
6. Compute selectivity estimates for a given selection and/or join operation.
7. Articulate how to modify an index structure to facilitate an index-only operation for a given relation.
8. For a given scenario decide on which indices to support for the efficient execution of a set of queries.
9. Articulate how DBMSs leverage parallelism to speed up query processing by dividing the work across multiple processors or nodes.

DM-Internals: DBMS Internals

This unit covers DBMS internals that are not directly involved in query execution.

KA Core:

1. DB Buffer Management (See also: SF-Resources)
2. Transaction Management (See also: PDC-Coordination:3)
    a. Isolation Levels
    b. ACID
    c. Serializability
    d. Distributed Transactions

3. Concurrency Control: (See also: OS-Concurrency)
    a. 2-Phase Locking
    b. Deadlocks handling strategies
    c. Quorum-based consistency models
4. Recovery Manager
    a. Relation with Buffer Manager

Non-Core:

5. Concurrency Control:
    a. Optimistic CC
    b. Timestamp CC
6. Recovery Manager
    a. Write-Ahead logging
    b. ARIES recovery system (Analysis, REDO, UNDO)

Illustrative Learning Outcomes

KA Core:

1. Articulate how a DBMS manages its Buffer Pool
2. Articulate the four properties for a correct transaction manager
3. Outline the principle of serializability

DM-NoSQL: NoSQL Systems

KA Core:

1. Why NoSQL? (e.g., Impedance mismatch between Application [CRUD] and RDBMS)
2. Key-Value and Document data model

Non-Core:

3. Storage systems (e.g., Key-Value systems, Data Lakes)
4. Distribution Models (Sharding and Replication) (See also: PDC-Communication:4)
5. Graph Databases
6. Consistency Models (Update and Read, Quorum consistency, CAP theorem) (See also: PDC-Communication:4)
7. Processing model (e.g., Map-Reduce, multi-stage map-reduce, incremental map-reduce) (See also: PDC-Communication:4)
8. Case Studies: Cloud storage system (e.g., S3); Graph databases ; “When not to use NoSQL” (See also: SPD-Web: 7)

Illustrative Learning Outcomes

KA Core:

1. Articulate a use case for the use of NoSQL over RDBMS.
2. Articulate the defining characteristics behind Key-Value and Document-based data models.

DM-Security: Data Security and Privacy

KA Core:

1. Need for, and different approaches to securing data at rest, in transit, and during processing.

2. Protecting data and database systems from attacks, including injection attacks such as SQL injection (See also: SEC-Security)
3. Database auditing and its role in digital forensics
4. Differences between data security and data privacy
5. Personally identifying information (PII) and its protection
6. Data inferencing and preventing attacks
7. Laws and regulations governing data security and data privacy (See also: SEP-???)
8. Ethical considerations in ensuring the security and privacy of data (See also: SEP-???)

Non-Core:

9. Typical risk factors and prevention measures for ensuring data integrity
10. Ransomware and prevention of data loss and destruction

Illustrative Learning Outcomes

KA Core:

1. Apply several data exploration approaches to understanding unfamiliar datasets.
2. Identify and mitigate risks associated with different approaches to protecting data.
3. Describe the differences in the goals for data security and data privacy.
4. Develop a database auditing system given risk considerations.
5. Describe legal and ethical considerations of end-to-end data security and privacy.

DM-Analytics: Data Analytics

KA Core:

1. Exploratory data techniques (e.g., Raj to fill in)
2. Data science lifecycle: business understanding, data understanding, data preparation, modeling, evaluation, deployment, and user acceptance
3. Data mining and machine learning algorithms: e.g., classification, clustering, association, regression (See also: AI-ML)
4. Data acquisition and governance
5. Data security and privacy considerations (See also: SEP-Security)
6. Data fairness and bias (See also: SEP-Security, AI-Ethics)
7. Data visualization techniques and their use in data analytics (Git-Ask Susan)
8. Entity Resolution

Illustrative Learning Outcomes

KA Core:

1. Describe several data exploration approaches, including visualization, to understanding unfamiliar datasets.
2. Apply several data exploration approaches to understanding unfamiliar datasets.
3. Describe basic machine learning/data mining algorithms and when they are appropriate for use.
4. Apply several machine learning/data mining algorithms.
5. Describe legal and ethical considerations in acquiring, using, and modifying datasets.
6. Describe issues of fairness and bias in data collection and usage

DM-Distributed: Distributed Databases/Cloud Computing

Non-Core:

1. Distributed DBMS (PDC-Communications)
    a. Distributed data storage
    b. Distributed query processing
    c. Distributed transaction model
    d. Homogeneous and heterogeneous solutions
    e. Client-server distributed databases (See also: NC-Introduction)
2. Parallel DBMS (See also: PDC-Algorithms)
    a. Parallel DBMS architectures: shared memory, shared disk, shared nothing;
    b. Speedup and scale-up, e.g., use of the MapReduce processing model (cross-reference CN/Processing, PD/Parallel Decomposition) (See also: SF-Basics:8)
    c. Data replication and weak consistency models (See also: PDC-Coordination)

DM-Unstructured: Semi-structured and Unstructured Databases

Non-Core:

1. Vectorized unstructured data (text, video, audio, etc.) and vector storage
    a. TF-IDF Vectorizer with ngram
    b. Word2Vec
    c. Array database or array data type handling
2. Semi-structured (e.g., JSON)
    a. Storage
        i. Encoding and compression of nested data types
    b. Indexing
        i. Btree, skip index, Bloom filter
        ii. Inverted index and bitmap compression
        iii. Space filling curve indexing for semi-structured geo-data
    c. Query processing for OLTP and OLAP use cases
        i. Insert, Select, update/delete trade offs
        ii. Case studies on Postgres/JSON, MongoDB and Snowflake/JSON

DM-SEP: Society, Ethics, and Professionalism

1. Issues related to scale (See also SEP-Economies)
2. Data privacy overall (See also: SEP-Privacy, SEP-Ethical-Analysis)
    a. Privacy compliance by design (See also: Privacy)
3. Data anonymity (See also: SEP-Privacy)
4. Data ownership/custodianship (See also: SEP-Professional-Ethics)
5. Reliability of data (See also: SEP-Security)
6. Intended and unintended applications of stored data (See also: SEP-Professional-Ethics)
7. Provenance, data lineage, and metadata management (See also: SEP-Professional-Ethics)
8. Data security (See also: SEP-Security)

Illustrative Learning Outcomes

1. Enumerate three social and three legal issues related to large data collections
2. Articulate the value of data privacy
3. Identify the competing stakeholders with respect to data ownership

4. Enumerate three negative unintended consequences from a given (well known) data-centric application (e.g., Facebook, LastPass, Ashley Madison)

Professional Dispositions

- **Meticulous:** Those who either access or store data collections must be meticulous in fulfilling data ownership responsibilities.
- **Responsible**: In conjunction with the professional management of (personal) data, it is equally important that data be managed responsibly. Protection from unauthorized access as well as prevention of irresponsible, though legal, use of data is paramount. Furthermore, data custodians need to protect data not only from outside attack, but from crashes and other foreseeable dangers.
- **Collaborative**: Data managers and data users must behave in a collaborative fashion to ensure that the correct data is accessed, and is used only in an appropriate manner.
- **Responsive**: The data that gets stored and is accessed is always in response to an institutional need/request.

Math Requirements

Required:

- Discrete Mathematics
    - Set theory (union, intersection, difference, cross-product)

Desirable Data Structures:

- Hash functions and tables
- Balanced (binary) trees (e.g., AVL, 2-3-4, Red-Black)

Course Packaging Suggestions

For those implementing a single course on Database Systems, there are a variety of options. As described in [1], there are four primary perspectives from which to approach databases:

- Database design/modeling
- Database use
- Database administration
- Database development, which includes implementation algorithms

Course design proceeds by focusing on topics from each perspective in varying degrees according to one's institutional context. For example in [1], one of the courses described can be characterized as design/modeling (20%), use (20%), development/internals (30%), and administration/tuning/advanced topics (30%). The topics might include:

- DM-SEP: Society, Ethics, and Professionalism (3 hours)

- DM-Data: The Role of Data (1 hour)
- DM-Core: Core Database System Concepts (3 hours)
- DM-Modeling: Data Modeling (5 hours)
- DM-Relational: Relational Databases (4 hours)
- DM-Querying: Query Construction (6 hours)
- DM-Processing: Query Processing (5 hours)
- DM-Internals: DBMS Internals (5 hours)
- DM-NoSQL: NoSQL Systems (4 hours)
- DM-Security: Data Security and Privacy (3 hours)
- DM-Distributed: Distributed Databases/Cloud Computing (2 hours)

Possibly, the more interesting question is how to cover the CS Core concepts in the absence of a dedicated database course. Perhaps the key to accomplishing this is to *normalize* database access. Starting with the introductory course students could be accessing a database versus file I/O or interactive data entry, to acquire the data needed for introductory-level programming. As students progress through their curriculum, additional CS Core topics can be introduced. For example, introductory students would be given the code to access the database along with the SQL query. By the intermediate level, they could be writing their own queries. Finally, in a Software Engineering or capstone course, they are practicing database design. One advantage of this *databases across the curriculum* approach is that allows for the inclusion of database-related SEP topics to also be spread across the curriculum.

In a similar vein one might have a whole course on the Role of Data from either a Security (SEC) perspective, or and Ethics (SEP) Perspective.

[1] The 2022 Undergraduate Database Course in Computer Science: What to Teach?. Michael Goldweber, Min Wei, Sherif Aly, Rajendra K. Raj, and Mohamed Mokbel. ACM Inroads, Volume 13, Number 3, 2022.

Committee

Chair: Mikey Goldweber, Denison University, Granville, OH, USA

Members:

- Sherif Aly, The American University in Cairo, Cairo, Egypt
- Sara More, Johns Hopkins University, Baltimore, MD, USA
- Mohamed Mokbel, University of Minnesota, Minneapolis, MN, USA
- Rajendra Raj, Rochester Institute of Technology, Rochester, NY, USA
- Avi Silberschatz, Yale University, New Haven, CT, USA
- Min Wei, Seattle, WA, Microsoft
- Qiao Xiang, Xiamen University, Xiamen, China