

# 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 data sets 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: 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 (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 developer) skills. These primarily include data modeling and query construction; be able 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:
  - o The Data Life Cycle:  
Creation-Processing-Review/Reporting-Retention/Retrieval-Destruction.
  - o The social/legal aspects of data collections: e.g. scale, data privacy, database privacy (compliance) by design, anonymity, ownership, reliability, 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 computing, MapReduce, 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.

### **Allocation of Core Hours**

| <b>Knowledge Unit</b>                             | <b>CS Core Hours</b> |
|---|----------------------|
| <b>The Role of Data</b>                           | 2                    |
| <b>Core Database Systems Concepts</b>             | 2                    |
| <b>Data Modeling</b>                              | 2                    |
| <b>Relational Databases</b>                       | 1                    |
| <b>Query Construction</b>                         | 2                    |
| <b>Query Processing</b>                           |                      |
| <b>DBMS Internals</b>                             |                      |
| <b>NoSQL Systems</b>                              |                      |
| <b>Distributed Databases/Cloud Computing</b>      |                      |
| <b>Semi-structured and Unstructured Databases</b> |                      |

A knowledge unit is labeled **CS Core (red)** or **KA Core (blue)** which encapsulates all the topics within the unit. If unlabeled, each topic is individually labeled. Finally, some knowledge units/topics are unlabeled and therefore considered “elective,” to indicate that while relevant, and an interesting component of this knowledge area, it is not considered a required knowledge unit. Learning outcomes are only enumerated for CS Core and KA Core knowledge units.

### **Description of Knowledge Units**

## The Role of Data (CS Core)

Topics:

- The Data Life Cycle:  
Creation-Processing-Review/Reporting-Retention/Retrieval-Destruction.
- The social/legal aspects of data collections:
  - scale
  - data privacy
  - database privacy (compliance) by design
  - anonymity
  - ownership
  - reliability
  - intended and unintended applications

Illustrative Learning Outcomes:

CS Core:

TBD.

## Core Database System Concepts (CS Core)

Topics:

- Purpose and advantages of database systems
- Components of database systems
- Design of core DBMS functions (e.g., query mechanisms, transaction management, buffer management, access methods)
- Database architecture, data independence, and data abstraction
- Use of a declarative query language
- Transaction mgmt
- Normalization
- Systems supporting structured and/or stream content (KA Core)
- Approaches for managing large volumes of data (e.g., noSQL database systems, use of MapReduce).
- Distributed databases/cloud-based systems
- Structured, semi-structured, and unstructured databases

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.

## Data Modeling

Topics:

- Data modeling
- Conceptual models (e.g., entity-relationship, UML diagrams) (KA Core)
- Spreadsheet models
- Relational data models (CS Core)
- Object-oriented models (cross-reference PL/Object-Oriented Programming)
  - GraphQL
- Semi-structured data model (expressed using DTD, XML, or JSON Schema, for example) (KA Core)
  - What is new in SQL:20??
- Specialized Data Modeling topics
  - Time series data (aggregation, and join)
  - Graph data (link traversal)
  - Materialized Views and Special data structures like (Hyperloglog, bitmap, ...)
    - Typically querying “Raw time series data” directly is very slow for things like “avg daily price”, “daily unique count”, “daily membership”
  - Geo-Spatial data

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:

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

## Relational Databases

Topics:

- Mapping conceptual schema to a relational schema (KA Core)
- Entity and referential integrity (CS Core)
  - Candidate key, superkeys
- Relational database design (CS Core)
- Physical database design: file and storage structures (KA Core)

- Functional dependency Theory (KA Core)
  - Closure of a set of attributes
- Normalization Theory
  - Decomposition of a schema; lossless-join and dependency-preservation properties of a decomposition (KA Core)
  - Normal forms (BCNF) (KA Core)
  - Denormalization (for efficiency) (KA Core)
  - Multi-valued dependency (4NF)
  - Join dependency (PJNF, 5NF)
  - 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:

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

## Query Construction

- Relational Algebra (KA Core)
- Relational Calculus
- SQL
  - Data definition including integrity and other constraints specification (KA Core)
  - Query formulation (CS Core)
  - Update sublanguage (KA Core)
- QBE and 4th-generation environments
- Different ways to invoke non-procedural queries in conventional languages (overlap with PL)
- Introduction to other major query languages (e.g., XPATH, SPARQL)
- Stored procedures

Illustrative Learning Outcomes:

CS Core:

1. Compose SQL queries that incorporate select, project, join, union, intersection, set difference, and set division.

## 2. Detect if a nested SQL query is correlated or not.

KA Core:

1. Compose queries in relational algebra that incorporate select, project, join, union, intersection, set difference, and set division.
2. Define, in SQL, a relation schema, including all integrity constraints and delete/update triggers.
3. Compose an SQL query to update a tuple in a relation.

## Query Processing (KA Core)

Topics:

- Index structures
  - B+ trees
  - Hash indices: static and dynamic
  - Index creation in SQL
- Algorithms for query operators
  - External Sorting
  - Selection
  - Projection; with and without duplicate elimination
  - Natural Joins: Nested loop, Sort-merge, Hash join
  - Analysis of algorithm efficiency
- Query transformations
- Query optimization
  - Access paths
  - Query plan construction
  - Selectivity estimation
  - Index-only plans
- Database tuning: Index selection
  - Impact of indices on query performance

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.

## DBMS Internals

Topics:

- DB Buffer Management (KA Core)
- Transaction Processing (KA Core)
  - Isolation Levels
  - ACID
  - Serializability
- Concurrency Control:
  - 2-Phase Locking (KA Core)
  - Deadlocks handling strategies (KA Core)
  - Optimistic CC
  - Timestamp CC
- Recovery Manager
  - Relation with Buffer Manager (KA Core)
  - Write-Ahead logging
  - ARIES recovery system (Analysis, REDO, UNDO)

Illustrative Learning Outcomes:

KA Core:

## NoSQL Systems

Topics:

- Why NoSQL? (e.g. Impedance mismatch between Application [CRUD] and RDBMS) (KA Core)
- Key-Value and Document data model (KA Core)
- Storage system (e.g. Key-Value system)
- Distribution Models (Sharding and Replication)
- Consistency Models (Update and Read, Quorum consistency, CAP theorem)
- Processing model (e.g. Map-Reduce, multi-stage map-reduce, incremental map-reduce)
- Case Studies: Cloud storage system (e.g. S3); Graph databases ; “When not to use NoSQL”

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.

## **Distributed Databases/Cloud Computing**

Topics:

- Distributed DBMS
  - Distributed data storage
  - Distributed query processing
  - Distributed transaction model
  - Homogeneous and heterogeneous solutions
  - Client-server distributed databases (cross-reference SF/Computational Paradigms)
- Parallel DBMS
  - Parallel DBMS architectures: shared memory, shared disk, shared nothing;
  - Speedup and scale-up, e.g., use of the MapReduce processing model (cross-reference CN/Processing, PD/Parallel Decomposition)
  - Data replication and weak consistency models

## **Semi-structured and Unstructured Databases**

Topics:

- Vectorized unstructured data (text, video, audio, ...) and vector storage
  - TF-IDF Vectorizer with ngram
  - Word2Vec [Word2vec - Wikipedia](#)
  - Array database or array data type handling
- semi-structured (e.g. JSON)
  - Storage
    - Encoding and compression of nested data types
  - Indexing
    - Btree, skip index, Bloom filter
    - Inverted index and bitmap compression
    - Space filling curve indexing for semi-structured geo-data
  - Query processing for OLTP and OLAP use cases
    - Insert, Select, update/delete trade offs
    - Case studies on Postgres/JSON, MongoDB and Snowflake/JSON

### **List of Professional Dispositions Appropriate for this KA**

- TBD

### **Math needed and wanted**

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

### **Shared Concepts**

- Society, Ethics, and Professionalism
- Modeling
- Programming Languages
- Systems Fundamentals
- Algorithms and Complexity
- Parallel and Distributed Computing
- Platform Based Computing
- Security/Information Assurance and Security
- Data Mining: See the ACM Data Science curriculum

### **Subcommittee**

Chair: Mikey Goldweber (Xavier University)

Subcommittee members

- Sherif Aly (The American University in Cairo)
- Sara More (John Hopkins University)
- Mohamed Mokbel (University of Minnesota)
- Rajendra Raj (Rochester Institute of Technology)
- Avi Silberschatz (Yale University)
- Min Wei (Microsoft)
- Qiao Xiang (Xiamen University)