

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 developer) 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:
 - 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, 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, 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 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:

Core Hours

Knowledge Unit	CS Core Hours	KA Core Hours
The Role of Data	2	
Core Database Systems Concepts	2	.5
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		
Total	9	25

A knowledge unit is labeled **CS Core (red)** or **KA Core (blue)** which encapsulates all the topics within the unit. 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

Knowledge Units

The Role of Data

Topics:

[CS Core - 2 hours]

- 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

Topics:

[CS Core - 2 hours]

- 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
- Approaches for managing large volumes of data (e.g., noSQL database systems, use of MapReduce).
- How to support CRUD-only applications
- Distributed databases/cloud-based systems
- Structured, semi-structured, and unstructured databases

[KA Core - .5 hours]

- 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.

Data Modeling

Topics:

[CS Core - 2 hours]

- Data modeling
- Relational data model

[KA Core - 3 hours]

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

[Elective]

- Spreadsheet models
- Object-oriented models (cross-reference PL/Object-Oriented Programming)
 - GraphQL
- What is new in SQL:202x
- 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 a conceptual data model.
3. Model a given environment using the document-based or key-value store-based data model.

Relational Databases

Topics:

[CS Core - 1 hours]

- Entity and referential integrity
 - Candidate key, superkeys
- Relational database design

[KA Core - 3 hours]

- Mapping conceptual schema to a relational schema
- Physical database design: file and storage structures
- Introduction to Functional dependency Theory
- Normalization Theory
 - Decomposition of a schema; lossless-join and dependency-preservation properties of a decomposition
 - Normal forms (BCNF)
 - Denormalization (for efficiency)

[Elective]

- Functional dependency Theory
 - Closure of a set of attributes
 - Canonical Cover

- Normalization Theory
 - 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

Topics:

[CS Core - 2 hours]

- SQL Query Formation

[KA Core - 4 hours]

- Relational Algebra
- SQL
 - Data definition including integrity and other constraints specification
 - Update sublanguage

[Elective]

- Relational Calculus
- 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. 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]

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

Query Processing

Topics:

[KA Core - 4 hours]

- 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:

[KA Core - 4 hours]

- DB Buffer Management
- Transaction Processing
 - Isolation Levels
 - ACID
 - Serializability
- Concurrency Control:
 - 2-Phase Locking
 - Deadlocks handling strategies
- Recovery Manager
 - Relation with Buffer Manager

[Elective]

- Concurrency Control:
 - Optimistic CC
 - Timestamp CC
- Recovery Manager
 - Write-Ahead logging
 - 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

NoSQL Systems

Topics:

[KA Core - 2 hours]

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

[Elective]

- 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.

Data Security & Privacy

Topics:

Tbd

Illustrative Learning Outcomes:

tbd

Data Analytics

Topics:

tbd

Illustrative Learning Outcomes:

tbd

Distributed Databases/Cloud Computing

Topics:

[Elective]

- 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:

[Elective]

- 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

Professional Dispositions

- **Professional:** This is potentially the most important disposition for Data Management. Though the notion of “data ownership” varies across national contexts, the importance of how an enterprise manages that data does not.
- **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:

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

Shared Concepts and Crosscutting Themes

Shared Concepts:

- Society, Ethics, and Professionalism
- Programming Languages
- Systems Fundamentals
- Algorithms and Complexity
- Parallel and Distributed Computing
- Specialized Platform Development
- Security
- Data Mining: See the ACM Data Science curriculum

Course Packaging Suggestions

Competency Specifications

- **Task 1:** Access data from a database for some purpose.
- **Competency Statement:** Access the database programmatically and iterate over the resulting relation performing some computation.
- **Competency area:** Software, Systems
- **Competency unit:** Development, testing
- **Required knowledge areas and knowledge units:**
 - DM / Query Construction
 - SEP / Social Context
 - SEP / Equity, Diversity and Inclusion
 - SEP / Methods for Ethical Analysis
 - SEP / Professional Ethics
- **Required skill level:** Develop
- **Core level:**

- **Task 2:** Produce a white paper assessing the social and ethical implications for collecting and storing the data from a new (or existing) application. (risk)
- **Competency Statement:** Identify the stakeholders and evaluate the potential long-term consequences for the collection and retention of data objects. Consider both potential harm from unintended data use and from data breaches.
- **Competency area:** Systems, Theory
- **Competency unit:** Evaluation, Management, Adaptation to social issues.
- **Required knowledge areas and knowledge units:**
 - SEP / Social Context
 - SEP / Methods for Ethical Analysis
 - SEP / Privacy and Civil Liberties
 - SEP / Professional Ethics
 - SEP / Security Policies, Laws and Computer Crimes
 - SEP / Equity, Diversity and Inclusion
 - DM / The Role of Data
 - SEC / Foundational Security
- **Required skill level:** Evaluate/Explain
- **Core level:**

- **Task 3:** Determine how to store a new application's data.
- **Competency Statement:** Choose and defend the appropriate data model (relational, key-value store, etc) for a given application.
- **Competency area:** Application, Theory

- **Competency unit:** Evaluation, Design
- **Required knowledge areas and knowledge units:**
 - DM / The Role of Data
 - DM / Core Database Systems Concepts
 - DM / Data Modeling
 - DM / Relational Databases
 - DM / NoSQL Systems
 - SEP / Security Policies, Laws and Computer Crimes
- **Required skill level:** Evaluate/Explain
- **Core level:**

- **Task 4:** Improve a database application's performance (speed)
- **Competency Statement:** Remove and/or create index structures to speed up the execution of frequent/important queries which are part of the application.
- **Competency area:** Software, Application
- **Competency unit:** Design, Improvement
- **Required knowledge areas and knowledge units:**
 - DM / Query Processing
 - DM / DBMS Internals
 - PD / Parallel Algorithms, Analysis, and Programming
- **Required skill level:** Evaluate, Develop
- **Core level:**

- **Task 5:** Secure data from unauthorized access
- **Competency Statement:** Create database views to ensure data access is appropriately limited.
- **Competency area:** Management
- **Competency unit:** Maintenance, Management
- **Required knowledge areas and knowledge units:**
 - DM / The Role of Data
 - DM / Relational Databases
 - DM / Query Processing
 - SEP / Security Policies, Laws and Computer Crimes
 - SEP / Professional Ethics
 - SEP / Privacy and Civil Liberties
 - SEC / Foundational Security
- **Required skill level:** Develop
- **Core level:**

- **Task 6:** Get back online after a disruption (e.g. power outage)
- **Competency Statement:** Restore the database to its most recent “safe” state using a recovery manager
- **Competency area:** Systems
- **Competency unit:** Maintenance, Management
- **Required knowledge areas and knowledge units:**
 - DM / DBMS Internals
- **Required skill level:** Learn how to use the appropriate restoration tool / Apply
- **Core level:**

- **Task 7:** Design the data storage for a new application
- **Competency Statement:** Model the application environment using an appropriate data model. If appropriate, convert to the relational model and normalize.
- **Competency area:** Application
- **Competency unit:** Design, Improvement
- **Required knowledge areas and knowledge units:**
 - DM / The Role of Data
 - DM / Data Modeling
 - SEP / Social Context
 - SEP / Methods for Ethical Analysis
 - SEP / Privacy and Civil Liberties
 - SEP / Professional Ethics
 - SEP / Security Policies, Laws and Computer Crimes
 - SEP / Equity, Diversity and Inclusion
 - SEC / Foundational Security
- **Required skill level:** Apply, Develop
- **Core level:**

- **Task 8:** Create a database for a new application.
- **Competency Statement:** Design the data storage needs (data modeling), assess the social and ethical implications for collecting and storing the data, determine how to store a new application’s data (RDBMS vs NoSQL), and create the database, including appropriate indices.
- **Competency area:** Design, Software, Systems
- **Competency unit:** Development, testing
- **Required knowledge areas and knowledge units:**
 - DM / The Role of Data
 - DM / Core Database Systems Concepts

- DM / Data Modeling
- DM / Relational Databases
- DM / NoSQL Systems
- DM / DBMS Internals
- SEP / Social Context
- SEP / Methods for Ethical Analysis
- SEP / Privacy and Civil Liberties
- SEP / Professional Ethics
- SEP / Security Policies, Laws and Computer Crimes
- SEP / Equity, Diversity and Inclusion
- SEC / Foundational Security
- **Required skill level:** Develop
- **Core level:**

Committee

Chair: Mikey Goldweber, Xavier University, Cincinnati, USA

Members:

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

Appendix: Core Topics and Skill Levels

KA	KU	Topic	Skill	Core	Hours
DM	The Role of Data	<ul style="list-style-type: none"> ● The Data Life Cycle ● The social/legal aspects of data collections: [crosslist SEP] 	Evaluate	CS	2
DM	Core DB System	<ul style="list-style-type: none"> ● Purpose and advantages of database systems ● Components of database systems ● Design of core DBMS functions (e.g., query mechanisms, transaction 			

	Concepts	<p>management, buffer management, access methods)</p> <ul style="list-style-type: none"> • Database architecture, data independence, and data abstraction • Transaction mgmt • Normalization • Approaches for managing large volumes of data (e.g., noSQL database systems, use of MapReduce). [crosslist PD] • Distributed databases/cloud-based systems • Structured, semi-structured, and unstructured databases 	Explain	CS	2
		<ul style="list-style-type: none"> • Use of a declarative query language 	Develop		
DM	Core DB System Concepts	<ul style="list-style-type: none"> • Systems supporting structured and/or stream content 	Explain	KA	.5
DM	Data Modeling	<ul style="list-style-type: none"> • Data modeling • Relational data models 	Develop	CS	2
DM	Data Modeling	<ul style="list-style-type: none"> • Conceptual models (e.g., entity-relationship, UML diagrams) • Semi-structured data model (expressed using DTD, XML, or JSON Schema, for example) 	Explain	KA	3
DM	Relational Databases	<ul style="list-style-type: none"> • Entity and referential integrity <ul style="list-style-type: none"> ◦ Candidate key, superkeys • Relational database design 	Explain	CS	1
DM	Relational Databases	<ul style="list-style-type: none"> • Mapping conceptual schema to a relational schema • Physical database design: file and storage structures • Functional dependency Theory • Normalization Theory <ul style="list-style-type: none"> ◦ Decomposition of a schema; lossless-join and dependency-preservation properties of a decomposition Normal forms (BCNF) ◦ Denormalization (for efficiency) 	Develop	KA	3

DM	Query Construction	<ul style="list-style-type: none"> • SQL Query Formulation 	Develop	CS	2
DM	Query Construction	<ul style="list-style-type: none"> • Relational Algebra • SQL <ul style="list-style-type: none"> ◦ Data definition including integrity and other constraints specification ◦ Update sublanguage 	Develop	KA	4
DM	Query Processing	<ul style="list-style-type: none"> • Index structures <ul style="list-style-type: none"> ◦ B+ trees ◦ Hash indices: static and dynamic ◦ Index creation in SQL • Algorithms for query operators <ul style="list-style-type: none"> ◦ 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 <ul style="list-style-type: none"> ◦ Access paths ◦ Query plan construction ◦ Selectivity estimation ◦ Index-only plans 	Explain	KA	4
		<ul style="list-style-type: none"> • Database tuning: Index selection <ul style="list-style-type: none"> ◦ Impact of indices on query performance 	Develop		
DM	DBMS Internals	<ul style="list-style-type: none"> • DB Buffer Management • Transaction Processing <ul style="list-style-type: none"> ◦ Isolation Levels ◦ ACID ◦ Serializability • Concurrency Control: [crosslist PD] <ul style="list-style-type: none"> ◦ 2-Phase Locking ◦ Deadlocks handling strategies • Recovery Manager <ul style="list-style-type: none"> ◦ Relation with Buffer Manager 	Explain	KA	4
DM	NoSQL System	<ul style="list-style-type: none"> • Why NoSQL? (e.g. Impedance mismatch between Application [CRUD] and RDBMS) • Key-Value and Document data model 	Explain	KA	2

DM	Data Analytics	tbd			
DM	Data Security & Privacy	tbd			