Final Exam Review

Lecture 16

Format

- 5-7 problems, with multiple sub-parts
- No notes, calculators, books, computers, phones, etc. may be used
- Your responses must be written in **pen**

Content

Everything, including…

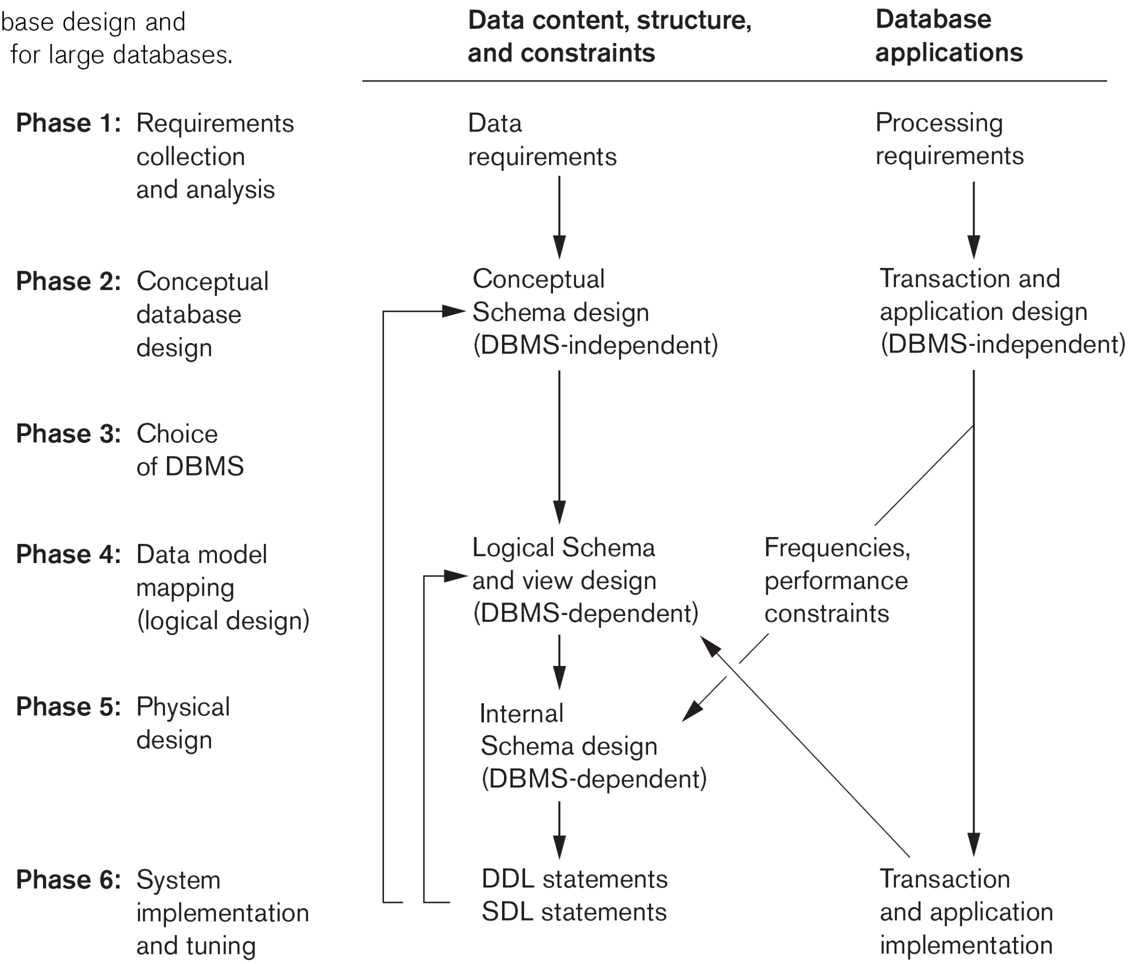
- General database knowledge
- The relational model
- SQL [programming]
- ER Diagrams
- Mapping ER Diagrams to Relations
- Normalization
- Physical design/tuning (incl. indexes)
- Database security
- Inverted index, r-tree

General Database Knowledge

- What is a transaction?
  - What are the properties that should hold for effective transaction processing?
- What is SQL?
  - DDL, DML
- What are the major phases of database design?
  - How do these apply to the material we've discussed: ERDs, relations, normalization, denormalization, indexes, views

Database Design & Implementation

Figure 10.1
Phases of database design and implementation for large databases.



Data content, structure, and constraints

Database applications

Phase 1: Requirements collection and analysis

Data requirements

Processing requirements

Phase 2: Conceptual database design

Conceptual Schema design (DBMS-independent)

Transaction and application design (DBMS-independent)

Phase 3: Choice of DBMS

Phase 4: Data model mapping (logical design)

Logical Schema and view design (DBMS-dependent)

Frequencies, performance constraints

Phase 5: Physical design

Internal Schema design (DBMS-dependent)

Phase 6: System implementation and tuning

DDL statements
SDL statements

Transaction and application implementation

Relational Model

- A database is composed of?
- A table schema is composed of?
- Each [schema component] has a ____ of valid _____ values?
- What is the difference between a set vs. bag of tuples?
  - In what context does each apply?
- Provide meaning/examples of each general category of constraints:
  - Implicit, Explicit, Application-based, Data Dependencies
- What kinds of constraints that can be defined in the schema?
  - What is a superkey vs. a key?
    - How do you identify a primary key? What happens to other super keys?
  - How do foreign keys fit in?

SQL [Programming]

- Know your terms/keywords: DDL/DML, `ASSERTION`/`TRIGGER`, `BEGIN`/`COMMIT`/`ROLLBACK`, `GRANT`/`REVOKE`/`WITH GRANT OPTION`, `REFERENCES`, `CLOB`/`BLOB`, ODBC, Impedance Mismatch, Result Set, Cursor, SQL Injection attack
- Given a schema and a query description, write SQL [to create, modify, query]
- Given SQL and a set of populated table(s), predict the result set

ER Diagrams & Mapping

- Conceptual design: goals, approaches
- All the notation we covered
  - Entities: weak/strong
  - Attributes: composite, multi-valued, derived, keys
  - Relationships: cardinality, structural, attributes
  - Specialization/Generalization
  - When to use!
- Mapping to tables
  - Multiple methods for specialization/generalization

Normalization

- What are the goals of normalization?
  - Spurious tuples? Additive decomposition?
  - Modification anomalies? Examples!
- Functional dependencies
  - Definition, relationship to keys
  - Trivial, transitive, full
- Normal forms
  - What do 1NF/2NF/3NF require?
  - Decomposition algorithm

Physical Design/Tuning

- Terms: clustered, covering, denormalization, [materialized] view, vertical/horizontal partitioning, EXPLAIN
- Compare B+-tree vs. hash table index
  - When would you use one vs. other?
- Given a description of a table (number of rows, attribute cardinalities) and a query, choose the appropriate index(es) to use – justify your choice
- Given a schema and a set of queries (with frequency/ execution time), develop an effective physical tuning plan (e.g. index[es], denormalization, partitioning, query rewriting)

Database Security

See all of lecture 13

- Great for terms/ideas
- Understand forms of authentication & access control, attack, mitigation tools, [techniques for] inferential security, and effective methods of encryption

Advanced Indexes

- Given a problem description, determine if an inverted index would be applicable; if so, provide a mapping to documents/words
- Given a problem description, determine if an r-tree would be applicable
- Given a set of documents, produce an inverted index
- Given an inverted index, full-text query
  - Should be able to handle logical or relational form of the index (e.g. what is the SQL to find all restaurants that take reservations? and take credit cards!?)