CS 338 Final Exam Review

LaTeXer: W. Kong

Chapter 1 : Intro

- **Metadata** = database definitions and information about the database
- Characteristics of the DB approach: (1) self-describing nature (2) insulation between programs and data + data abstraction (3) multiple views of the data (3) sharing of data and multi-user transaction processing
- **Program-data independence** = structure of how data is stored independent from programs that access it
- **Program-operation independence** = independence of function calls relative to their implementation
  - Data abstraction allows the above two
- **Conceptual representation of data** = no details of how data is stored or implementation of operations
  - E.g. A data model may be used as a conceptual representation
- **View** = subset of data that is derived from data of the DB but is not explicitly stored
- Transactions = **isolated** (one at a time) and **atomic** (all or nothing)
  - DBA = authorize, coordinate, acquire; DB designer = identify, choose stuctures;
  - Sys. Anlyst = determine reqs.; App progrmrs = implement through progs.
  - End users - casual, naive/parametric, sophisticated, standalone
- Advantages? [De] Normalization, redundancy control, security, backup and recovery, GUIs, relationship representation
- Types of **Integrity Constraints**: (1) Referential (dependent existence) (2) Key/Uniqueness (3) Business Rules (4) Inherent data model rules (e.g. NULLs)
- **Inferencing**: deductive DB systems (ad hoc), triggers (happens after an update), stored procedures

Chapter 2 : DBMS

- Client/Server architecture ˜ Chef/Customer relationship
- Data Models = collection of concepts that describe a database's structure
  - allows data abstraction, basic operations, and set state validity
  - 3 categories: low/**physical** (detailed) [uses access paths and indices], high/**conceptual** (usual) and **representational** (end users)
  - can also be split between relational and object-oriented
- **3-Schema Architecture** == (1) Internal = physical description (2) Conceptual = describe whole DB (3) External/View = selective subset for end-users
- **Data Independence** = Capacity to change schema at one level of DBN without changing the next; Split into logical vs. physical
- **Tiered architectures** = extra layers between client and server communication
- DBMS classification criteria = data model, number of users, number of sites, access paths, cost
- Other slides are brought up again in other chapters

Chapters 3 to 5 : Keys + SQL

- Degree = number of attributes in a relation
- Cardinality = total number of values in domain
- Understand the following notation: $R(A_1, A_2, A_3)$, $v_i = t[A_i] = t.A_i$
- **Superkey** = distinct identifier; **Key** = minimal superkey = **candidate key**, **Primary key** & **Unique key** $\in$ Keys, **Foreign keys** are superkeys (they require total participation)
- More **Integrity constraints** = (1) Functional ICs, (2) Semantic ICs, (3) State ICs (4) Transition ICs
- *We will ignore the section on SQL and ER/EER diagrams as these have been well studied for the midterm and during assignments*

Chapter 6 : Relational Algebra

- $A \leftarrow \sigma_{<cond.>}(R)$ = partitions the relation via subsetting tuples (horizontal partition)
- $A \leftarrow \pi_{<attr.\ lst.>}(R)$ = partitions the relation via subsetting attributes (vertical partition)
- $A \leftarrow \rho_{(B_1,\ldots,B_n)}(R)$ = explicit rename of attributes to the $B_i's$
- $A \leftarrow R \bowtie_{<cond.>} Q$ = joins R and Q based on the condition $< cond. >$
- Other operations are $\cup, \cap, \times, -$

Chapters 7 to 10 : Mapping Algorithms

- **9-step mapping algorithm** = (1) map regular entities (2) map weak entities (3) map 1:1 relations (4) map 1:N relations (5) map M:N relations (6) map multi-valued attributes (7) map N-ary relationships (8) map generalized and specialized relations (9) map union types
- **IS Life Cycle** = (1) Feasibility analysis (2) Requirement collection and analysis (3) Design (4) Implementation (5) Validation and Acceptance testing (6) Deployment, operation and maintenance
- **Design Phase** = (1) Requirement collection and analysis (2) Conceptual DB design (3) Choice of DBMS (4) Data model mapping (5) Physical DB design (6) DB system implementation and tuning
- **UML** (see MS Visio) = split into structural and behavioural; can be used to design relational or object oriented DBs
  - Structural types include (1) class diagrams / package diagrams (2) object diagrams (3) component diagrams (4) deployment diagrams
  - Behavioural types include (1) use case diagrams (2) sequence diagrams (3) collaboration diagrams (4) statechart diagrams (5) activity diagrams
- **Semantic web** = helps to interpret the meaning of the data
  - Could looks at usage, areas of the data that are the most use

Chapters 12 to 14 : XML & PHP

- Excluded is: Part of extracting XML Documents from Relational Databases; writing out PHP forms is excluded but exam will require a general ability to understand PHP variable names
- **Well formed** in XML is when the XML version is included, all elements are in a start/end tag container and everything is encapsulated in one large container
  - **Valid** in XML is well formed, follows a schema, and a structure in a separate XML DTD file
- **XPath** in XML uses the * wild card symbol with @ as an attribute declaration
- **PHP** uses $ to create variable names
- **Arrays** come in either numeric (standard) or associative (like a dictionary)
- Programming conventions = Some examples are **persistent stored modules** (PSM), **embedded SQL** (API), or **database programming approach** (from scratch)

Chapter 15 : Functional Dependency

- $X \leftarrow Y$ if $t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y]$
- **1NF** = does not allow nested relations; create new relations for complex attributes
- **2NF** = dependency (non-prime attributes) must be from ALL of a primary key (i.e. not PART of a primary key); also any non-prime attribute cannot be dependent on ANY key on the relation
- **3NF** = 2NF + transitive dependency is not allowed; this is because we can have two tuples with different PKs but same values for a single attribute; if the attribute (with the same values) determine another attribute, there will be a problem (e.g. dname depends on dnum but multiple same values of dnum; 2NF is satisfied if we have only one PK)
  - Also defined in the notes; go check the notes!
  - Can allow transitive dependency if the final attribute in the dependency is a prime attribute (attribute that is part of any candidate key)
- **BCNF** = 3NF minus the fact that we can have the final attributes in a transitive dependency as a prime attribute
- A trivial dependency is when an attribute determines itself
- **Anomaly types** = Insertion, Deletion and Modification

- **Spurious tuples** = excessive or invalid information (extra attributes or too many NULLs) created by joins

Chapter 21 : Transaction Processing

- **Granularity** = Size of a data item in a database
- **Lost update rule** = when two transactions update the same data item, but both read the same original value before update
- **The Temporary Update** (or Dirty Read) **Problem** = when one transaction T1 updates a database item X, which is accessed (read) by another transaction T2; then T1 fails for some reason ; X was (read) by T2 before its value is changed back after T1 fails
- **The Incorrect Summary Problem** = one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records (some entries are not entirely updated)
- **The Unrepeatable Read Problem** = a transaction T1 may read an item; later, T1 may read the same item again and get a different value because another transaction T2 has updated the item between the two reads by T1
- **Causes of transaction failure** = (1) computer failure/system crash [hardware/software] (2) a transaction/system failure [logic] (3) local errors/ exception conditions [programmed errors] (4) concurrency control [deadlock/abortion] (5) disk failure (6) physical problems/catastrophes
- **Transaction states** = (1) active (2) partially committed (3) committed (4) failed (5) terminated
- **ACID** = Atomicity, Consistency, Isolation, Durability/Permanency (Recoverability)
  - The consistency preservation property of the ACID properties states that: each transaction if executed on its own (from start to finish) will transform a consistent state of the database into another consistent state
- **Recoverable schedule** = no committed transaction needs to be rolled back; also if no transaction T in S (a schedule) commits until all transactions T' that have written an item that T reads have committed
- **Cascadeless schedule** = A schedule in which a transaction T2 cannot read an item X until the transaction T1 that last wrote X has committed
  - The set of cascadeless schedules is a subset of the set of recoverable schedules
- **Strict schedule** = A schedule in which a transaction T2 can neither read nor write an item X until the transaction T1 that last wrote X has committed
  - Blind write = A write operation w2(X) that is not preceded by a read r2(X).
  - If blind writes are not allowed, all cascadeless schedules are also strict
  - The set of strict schedules is a subset of the set of cascadeless schedules
- **Serial schedules** are not feasible for performance reasons:
  - No interleaving of operations
  - Long transactions force other transactions to wait
  - System cannot switch to other transaction when a transaction is waiting for disk I/O or any other event
  - Need to allow concurrency with interleaving without sacrificing correctness
- **Result equivalent =** two schedules are called result equivalent if they produce the same final state of the database
- **Conflict equivalent** = two schedules are conflict equivalent if the relative order of any two conflicting operations is the same in both schedules
  - A schedule S is said to be serializable if it is conflict equivalent to some serial schedule S'
  - Construct precedence graph (from early to later) between R/W, W/R, W/W conflicts (cycles imply non-serializability)
- **View equivalence** = conflict serializability when blind writes are allowed
  - A schedule is view serializable if it is view equivalent to a serial schedule
  - The same set of transactions participates in S and S'
  - For any operation Ri(X) of Ti in S, if the value of X read was written by an operation Wj(X) of Tj (or if it is the original value of X before the schedule started), the same condition must hold for the value of X read by operation Ri(X) of Ti in S'
  - If the operation Wk(Y) of Tk is the last operation to write item Y in S, then Wk(Y) of Tk must also be the last operation to write item Y in S'

- **Phantoms** = New row inserted after another transaction accessing that row was started.

Chapter 22 : Concurrency

- **Purpose of Concurrency Control** = (1) To enforce Isolation (through mutual exclusion) among conflicting transactions (2) To preserve database consistency (3) To resolve read-write and write-write conflicts
- **Read/Write locks**:

	Read	*Write*
Read	*Y*	*N*
Write	*N*	*N*

where $Read \uparrow Write$ and $Write \downarrow Read$

- A transaction is **well-formed** if (1) It must lock the data item before it reads or writes to it. (2) It must not lock already locked data items and it must not try to unlock a free data item.
- **Two-phase locking**: *Locking (Growing) Phase*: locks applied on data items → *Unlocking (Shrinking) Phase*: data items unlocked [MUST BE **MUTUALLY EXCLUSIVE**]
  - **Conservative**: Prevents deadlock by locking all desired data items before transaction begins execution
  - **Basic**: Transaction locks data items incrementally
  - **Strict**: Write unlocking is performed after a transaction terminates
  - **Rigorous**: Read and write unlocking is performed after a transaction terminates
- **Wait for graph** = like a precedence graph; new waits/points for/to old
- **Timestamps** = to allow concurrency but to resolve as many problems and conflicts as possible
  - When a transaction (R/W) wants to execute under a smaller timestamp than the highest recorded one, it is rolled back and executed with a higher timestamp
  - **Thomas' write rule** = read_ TS(X) > TS(T) and T is rolled back, the most recent operation should be rejected; in consecutive writes, ignore the earlier one
- **Starvation** = occurs when a particular transaction consistently waits or restarted and never gets a chance to proceed further
  - **Wound-wait:** younger aborts, older continues
  - **Wait-die:** same as the above but the older one waits
- **Certify lock** = allows the holding off of R/W in order to create an "official" version of a variable; done before committing
  - Used in a system where reads are done on the latest "version" of a variable changed via a write; writes create new "version"
- **Optimistic Concurrency Control Schemes**
  - **(1) Read phase** = read committed data values; updates on local data versions
  - **(2) Validation phase**
    - Tj completes its write phase before Ti starts its read phase
    - Tj completes its write phase before Ti starts its write phase and the read_set_Ti ∩ write_set_Tj = ∅
    - the read_set and write_set of Ti have nothing in common with write_set of Tj
  - **(3) Write phase** = On a successful validation transactions' updates are applied to the database; otherwise, transactions are restarted.

Chapter 23 : Recovery

- **Dirty bit** (0 or 1) (1) = Associated with each block (2) Indicates if the buffer has been modified
- **Pin/unpin bit** (0 or 1) = (1) Associated with each block (2) Indicates if the buffer can (or cannot) be written back to the disk
- **Steal** = (1) Uses pin-unpin bit (2) Steal approach allows writing an updated block to DB on disk by uncommitted transactions
- **Force approach** = If ALL pages modified by T are immediately written to disk before T commits
  - No Force + Steal is ideal:

No force		*Desired*
Force	*Trivial*	
	No steal	*Steal*

- The recovery subsystem may maintain the following lists: (1) Active lists (undo-list) (2) Committed lists (redo-list) (3) Aborted lists
- To prevent rollback, **Strict 2-Phase Locking** is enforced to guarantee (1) Recoverability (2) Cascadeless (3) Efficient rollback

- **Deferred update** = The DB can be updated only by committed transactions; NO-UNDO + REDO
  - Maintain two lists of transactions (1) Committed Transactions (CT) since the last checkpoint (2) Active Transactions (AT) (AT)
  - REDO all write operations of Committed Transactions from LOG in the order in which they were written into the log
  - Ignore ALL Active Transactions and restart them
- **Immediate update** = The DB may be updated by uncommitted transactions
  - If all updates of a transaction T are recorded in the DB on the disk before T commits, then NO-REDO is needed (UNDO + NO-REDO)
  - If T must commit before it can apply all of its changes to DB, then we have (UNDO + REDO)
  - Maintain AT and CT and UNDO all the write_items operations of AT backward from log
  - REDO all the write_items operations of CT forward from log

Chapter 24 : Security

- Types of **security counter-measures** = (1) Access control (2) Inference control [deduction prevention] (3) Flow control [consistent security levels] (4) Encryption
- **Threat groups** = (1) Murphy's Law (2) Amateurs (3) "Script kiddies" (4) Crackers (5) Organized crime (6) Government "cyberwarriors" (7) Terrorists / Activists
- "A system is only as strong as its weakest link" + "Security is economics"
- **Issues of security** = (1) Legal and Ethical (2) Policy (3) System related (4) The need to identify various security levels
- **Threats to DBs** = loss of (1) Integrity [students change grades] (2) Availability (3) Confidentiality
- **Defense tactics** = (1) Prevent (2) Deter (3) Deflect (4) Detect (5) Recover/Mitigate
- **Mechanisms** = Discretionary vs. Mandatory [Default]
- **DBAs** can ... (1) Create accounts [Access Control] (2) Grant privileges [Discretionary] (3) Revoke Privileges [Discretionary] (4) Assign security levels [Mandatory]
- **DB Audits and Audit Trails** [Access Control]
  - The database system must also keep track of all operations on the database that are applied by a certain user throughout each login session via the system log
  - If any tampering with the database is suspected, a database audit is performed
  - A database log that is used mainly for security purposes is sometimes called an audit trail.
- Typical **security classes** are top secret (TS), secret (S), confidential (C), and unclassified (U), where TS is the highest level and U the lowest: $TS \geq S \geq C \geq U$
- Two restrictions are enforced on data access based on the **subject/object classifications**:
  - **Simple security property**: A subject S is not allowed read access to an object O unless class(S) $\geq$ class(O).
  - A subject S is not allowed to write an object O unless class(S) $\leq$ class(O). This known as the **star property** (or * property).
- A multilevel relation will appear to contain different data to subjects (users) with different clearance levels.
  - In some cases, it is possible to store a single tuple in the relation at a higher classification level and produce the corresponding tuples at a lower-level classification through a process known as **filtering**.
  - In other cases, it is necessary to store two or more tuples at different classification levels with the same value for the **apparent key**.
  - This leads to the concept of **polyinstantiation** where several tuples can have the same apparent key value but have different attribute values for users at different classification levels.
- **Truman semantics**: the DBMS pretends that the data the user is allowed to access is all the data there is
- **Non-Truman semantics**: the DBMS can reject queries that ask for data the user is not allowed to access
- **Statistical Inference** example attack: SELECT SUM(salary);SELECT SUM(salary) WHERE lastname != 'Adams'
- **Flow control** regulates the distribution or flow of information among accessible objects.
  - A **flow** between object X and object Y occurs when a program reads values from X and writes values into Y.
  - A **flow policy** specifies the channels along which information is allowed to move.

- A **covert channel** allows a transfer of information that violates the security or the policy and information to pass from a higher classification level to a lower classification level through improper means (confidential vs. non-confidential channels)

Chapter 25 : Distributed Databases

- **Distributed Database** = A logically interrelated collection of shared data (and a description of this data), physically distributed over a computer network
- **Distributed DBMS (DDBMS)** = Software system that permits the management of the distributed database and makes the distribution transparent to users
  - **Advantages**? **[A]** We can decentralize processing AND logically integrate information **[B]** We can create multiple levels of transparency: (1) physical (2) logical (3) distribution/network (4) network (5) naming (6) replication (7) fragmentation **[C]** There is increased reliability, performance, scalability and availability
  - **Disadvantages?** **[A]** Complexity, Cost, Security, Integrity control, Lack of standards **[B]** Concurrency Control and Recovery Issues
- **Transparency** = hide information from the user; implemented through logical/physical independence
- **Autonomy** = to what degree databases in a connected distributed database can operate independently
- Different **types of DDBMS's**
  - Homogeneous = all sites have identical setup which is much easier to manage
  - Heterogeneous = sites may have different data models, DBMS products, etc.
    - Federated = different DB systems but managed through a single conceptual schema
    - Multi-database = for data access, the schema is constructed dynamically as needed
    - Semantic = semantic meaning differs across various systems
- **DDBMS Design** = (1) Fragmentation (Horizontal, Vertical, Mixed) (2) Data Allocation (3) Replication
- **Fragmentation** allows = (1) Locality of Reference (2) Improved Reliability, Availability, Performance and Security (3) Balanced Storage Capacities and Costs (4) Minimal Communication Costs (5) Parallelism
  - Disadvantages are (1) Performance in aggregation (2) IC Propagation
- **Data allocation** = (1) Centralized (2) Partitioned (3) Complete Replication (4) Selective Replication
- **Data Replication =** (1) fully (2) partially (3) no replication
- Problems of **Concurrency Control and Recovery**
  - (1) multiple copies of data items (2) failure of individual sites (3) communication link failure (4) distributed commit (by fragmentation) (5) distributed deadlock
  - **Primary Site Method** = One site is authoritative in backing up and creating official reads (easy to backup)
    - Concurrency control and commit are managed by this site; In two phase locking, this site manages locking and releasing data items
    - Could be overloaded or danger of inaccessible data if it goes down
  - **Primary Copy Method** = Each data source has one site with authority (hard to backup)
    - Identification and aggregation is complex
  - In both approaches a coordinator site or copy may become unavailable. This will require the selection of a new **coordinator**
  - Primary and backup sites fail or no backup site: Use **election process** to select a new coordinator site; can be used for lock grants with a time-out option