Comp Sci 564 – fall 2019

Exams

1. midterm: take home 10/23, due 11/1 noon – value 15% – instructor and all TAs grade it by 11/11.
2. final: 12/13 10:05am–12:05pm – location tbd – value 30% – instructor and all TAs grade it as needed by the university – final grades are due 12/19

Review sessions – regular class time & place

1. Mon 10/21 + Wed 10/23 – before midterm
2. Fri 11/22 – practice final exam
3. Wed 11/27 – before Thanksgiving
4. Fri 12/6 – before final (instructor)
5. Wed 12/9 – before final (TAs)

Projects – single person

In addition, there will be reading assignments

1. assigned 9/6, due 9/15 – value 10% – parse and locate, C++ warm up – Zhihan runs the project, Ruohui grades it by 9/22
2. assigned 9/20, due 9/30 – value 10% – write SQL queries – Kyle runs the project, Ruohui grades it by 10/6
3. assigned 10/4, due 10/20 – value 20% – code b-trees in C++ – Kyle runs the project, all TAs grade it by 11/3 (Halloween on 10/31)
4. assigned 11/15, due 11/22 – value 15% – rewrite SQL queries – Zhihan runs the project, all TAs grade it by 12/8 (Thanksgiving on 11/28)

Lectures & in-class discussions

1. Introduction (9/4)
    a. sharing structured data: cleaning, schema (constraints), logical and physical data independence, security, transactions, availability, analytics
    b. neighboring areas within computer science: operating systems, file systems, storage, networking, communication, programming languages, compilers, theory & complexity, data structures & algorithms, high-performance & super computing, fault-tolerance, software engineering, hardware, system/data center designs
    c. system architecture
        i. process manager: startup/shutdown, failover, login/authentication, admission control/dispatch, memory manager, load management, monitoring & admin
        ii. communication & client library, client-side caching & coherency
        iii. relational engine (data definition & catalogs, parsing, views, security, rewrite, query optimization, query execution, index tuning)

        iv. storage engine (I/O, buffer pool, access methods, concurrency control & locking, write-ahead logging & recovery)
        v. utilities (e.g., backup & restore, consistency check & repair, index creation & defragmentation, loading, replication & log shipping)

2. Database design (9/9)
    a. ER diagrams: entity types (strong, weak), relationship types (many-to-many, many-to-one, identifying), attributes (identifying, complex, multi-valued)
    b. Tables, rows, columns: mapping ER diagrams to tables, primary keys, foreign keys
    c. Database theory: functional dependencies, normal forms, universal relation
3. SQL (9/11+16)
    a. create table
        i. column types: int, numeric, date, datetime, char, varchar
        ii. integrity constraints: primary key, foreign key, unique, not null, check
    b. DML: insert, update, delete, select
    c. query clauses: select, from, where, order by, distinct
    d. relational algebra, default mapping from SQL to RA, Cartesian products
    e. group by, having, nested queries, views, "with"
    f. RA in the "from" clause: (select), outer & semi joins, set operations
    g. permissions
4. Memory, storage, access, search (9/18)
    a. storage devices & memory hierarchy
        i. traditional disk drives, tracks, pages, random vs sequential access
        ii. disk arrays, RAID-4/5/6, log-structured file systems
        iii. flash devices, block erasure, flash translation layer
        iv. buffer pool admission & replacement, pinning, keep/toss
        v. CPU caches, admission, replacement policies & mechanisms
    b. Search in arrays
        i. sequential, binary, interpolation, extrapolation
        ii. complexity vs efficiency vs robust performance
5. B-trees (9/23)
    a. binary search in paged arrays: cache, imprecise center, array stretch & holes
    b. tree search & node search, invariants (depth)
    c. index creation, key insertion, key deletion, key update, non-key update
    d. node organization, size optimization, prefix truncation, fence keys for self-testing
    e. node split heuristics: center, fill factor, suffix truncation
    f. primary and secondary indexes, uniqueness, included columns
    g. automatic index selection (tuning), maintenance, selection (query optimization)
    h. key construction: hash values, versions, space-filling curves, merged indexes
    i. normalized keys, poor man's normalized keys
    j. compression: prefixes, lists of row identifiers, bitmap indexes, run-length encoding
    k. for postgres b-trees, see here; for foster b-trees, see here.
6. External merge sort (9/25 – guest speaker: Thanh Do)
    a. usage: b-tree creation, "order by", "distinct", "group by", top/limit, joins, set ops
    b. run generation: quicksort, priority queues, memory management, (in-memory index)
    c. merging, merge plans, merge optimizations (smallest runs)
        i. read-ahead: double-buffering, forecasting
        ii. cliffs: binary switch vs incremental transition, graceful degradation
        iii. page size optimization: try to beat page size = latency × bandwidth

    d. parallel sorting: shuffle/exchange before, during, or after sorting

7. Query execution algorithms (9/30 + 10/2)
    a. table scan ("a track at a crack"), column scans, index scans, index intersection
    b. duplicate removal: in-stream, in-sort, hash-distinct
    c. grouping: in-stream, in-sort, hash aggregation
    d. nested loops join: naive, block, index, temporary index, poor man's merge join
    e. nested queries: nested iteration, caching, side-way information passing
    f. merge join: duplicate key values, zigzag, sort on normalized keys or hash values
    g. hash join: internal, external, recursive, graceful degradation
    h. adaptive join: cliff vs graceful degradation (see here and here)
    i. set operations: intersection, union, difference
    j. join within sort-aggregation, generalized aggregation, generalized join
    k. complex plans: execution algebra, iterators, bulk API, memory management
    l. parallel algorithms & execution
        i. exchange (a heap in each leap)
        ii. semi-join reduction, bit vector filtering, graceful degradation in lookup join
        iii. "best-effort" in-memory-only opportunistic pre-aggregation
    m. plan & predicate compilation & inlining, vectorized execution, normalized keys
    n. narrow and wide update plans, foreign-key integrity constraints (restrict, cascade), maintenance of indexed views, deferred updates, write-optimized storage & b-trees, adaptive merging & update propagation
8. Metadata = database catalogs (10/7)
    a. ER diagrams: database, table, integrity constraint, permission, column, index, view, query, user/role
    b. caching – swizzling pointers or entire data structures
    c. integrity constraints: catalogs, tests, enforcement, cascading
    d. Histograms & synapses, types & information contents & creation effort, unique values (e.g., of index prefixes)
9. Query optimization (10/9-16)
    a. history: System R, Ingres, Kooi, RTI, Almaden, Oracle, SQL Server, Greenplum
    b. cardinality estimation: constraints, histograms, "magic numbers" – correlations, distinct key values & groups, regular expressions on strings – sampling (dynamic, materialized samples), machine learning
    c. cost calculation: network, I/O, CPU, row counts – robustness
    d. expression & plan enumeration, plan selection – expected cost, robustness
    e. plan compilation – System R, Ingres, Hyper
    f. example rewrites & optimizations
        i. pushing down selection and projection
        ii. join order – dynamic programming vs incremental transformations
        iii. equivalence classes, inferred predicates
        iv. interesting orderings
        v. pushing down grouping operations
        vi. functional dependencies in "order by" and "group by" – group-join
        vii. nested queries ⇒ semi-joins, (local) rewrite – caches
        viii. star join plans: semi-join reduction, Cartesian products
        ix. "count (distinct) > 1"
        x. pivot operations?
        xi. SQL Server query plans

10. ACID transactions – brief intro (10/18)
11. Review sessions (10/21+23) – class cancelled (10/25)
12. Write-ahead logging, transaction rollback, traditional restart, instant restart (10/28+30)
13. Concurrency control (11/4+6)
    a. ACID transactions: atomicity (indivisibility, "all or nothing"), consistency, isolation (concurrency control, synchronization atomicity), durability (persistent after commit)
    b. low-level vs high-level concurrency control, latches vs locks
    c. locking in indexes: key-range locking, key-value locking
    d. optimistic concurrency control, timestamps
    e. snapshot isolation, multi-version storage
14. Bitmap indexes, hash indexes, master-detail clustering, blob storage in databases, key-value stores (11/8)
15. Class cancelled (11/11)
16. Column storage, compression (dictionary, run-length encoding), zone maps, zone filters, zone indexes, late materialization, write-optimized storage (11/13)
17. Log-structured merge-forests, stepped-merge forests, partitioned b-trees, compaction policies, bit vector filtering (11/15)
18. Scaling & parallel query execution (11/18)
    a. cores and hyperthreading
    b. shared everything (shared memory), shared disk (shared storage), shared nothing
    c. partitioning a workload, table, or index
    d. parallel query execution – pipelining and partitioning – exchange iterator
    e. log shipping vs mirroring
    f. two-phase commit, read-only optimization, delegation
    g. tiers in database management & storage, log shipping to storage
19. Cloud services (11/20)
    a. efficiency, resource isolation, metering, billing
    b. tiers: applications, relational engine, transactional access methods, storage – shared-disk technologies
    c. self-management & -tuning & -repair, scalable management
20. Project 4 (query tuning) and final exam (practice) (11/22)
21. In-memory databases (11/25)
    a. predicate compilation, lock-free data structures, rows vs columns, pointer swizzling
    b. write-ahead logging, repair & restart & restore & failover, mirroring & log shipping
    c. remote commit, log shipping to storage
    d. the five-minute rule(s)
22. Review session (11/27 – Wednesday before Thanksgiving) – no class 11/29

possible further sessions: 12/2+4+6

23. Hardware support: FPGA (filters, decompression), sorting, RDMA
24. Data cleaning: spelling, normalization, duplicate detection, entity resolution
25. Data analytics: business intelligence, machine learning
26. Review sessions (12/9+11) – last sessions before the final exam – TAs
27. Final exam (Friday 12/13)

Conceivable exam questions

Also review the midterm exam and the practice final exam, including their sample answers.

1. If databases and database management systems (DBMS) are all about sharing structured data, what are some (at least three) required or highly desirable aspects of that?
   Metadata (catalogs, information describing the data) – logical and physical data independence (e.g., separation of tables and indexes) – transactions (atomicity, consistency, isolation, durability).
2. What are some (at least three) of the subject areas within computer science neighboring database management?
   Programming languages, operating systems, user interfaces, hardware accelerators.
3. What is part of the storage engine (SE) within a DBMS, what is part of the relational engine (RE)? (Name at least two components each.)
   SE: indexing, backup and restore, concurrency control, logging, and recovery.
   RE: SQL parser, catalogs, query optimization, query execution.
4. How is an entity type within an ER-diagram mapped to a relational DBMS?
   An entity type becomes a table.
5. How is a relationship type within an ER-diagram mapped to a relational DBMS?
   A many-to-many relationship becomes a table with two foreign keys; a many-to-one relationship becomes a foreign key on the many-side.
6. How is a multi-valued attribute within an ER-diagram mapped to a relational DBMS?
   It becomes a weak entity type, i.e., a separate table with a foreign key integrity constraint.
7. What five kinds of integrity constraints are supported within DBMSs? (Name at least three kinds.)
   Check, not null, unique, primary key, foreign key.
8. What is required within a database to efficiently support a non-null constraint?
   A simple predicate is applied during all updates.
9. What is required within a database to efficiently support a uniqueness constraint?
   An index for efficient lookup during updates.
10. What happens if a user transaction deletes a primary key if there is a foreign key constraint specified with "on delete restrict"?
    If there is a row with a foreign key value equal to the primary key value being deleted, the update fails.
11. What happens if a user transaction deletes a primary key if there is a foreign key constraint specified with "on delete cascade"?
    If there is a row with a foreign key value equal to the primary key value being deleted, these rows with that foreign key value will also be deleted.
12. Write a "create table" SQL statement for an entity type, e.g., a machine part that an industrial company or a retailer might stock in their warehouse. Include at least three kinds of integrity constraints.
    create table part (part_id int primary key, size float not null check (value > 0))
13. Write a "create table" SQL statement for a relationship type. Include at least three kinds of integrity constraints.
    create table enrollment (student_id int not null foreign key references student, course int not null foreign key references course, grade numeric check (value >= 0), primary key (student_id, course))
14. For a table created using "create table student (identifier int primary key, name varchar(50) not null, class_of int check (value > 2010))", write a query that produces names of students who are overdue to graduate, i.e., their class_of date is in the previous year or earlier.
    select * from student
    where datepart ('year', class_of) < datepart ('year', today ())

15. For the query of the last question, write or draw an expression in relational algebra.
    …
16. For the query of the last question, write or draw a query execution plan.
    … perhaps using a table scan
17. For the query of the last question, write or draw an alternative query execution plan.
    … perhaps using an index on class_of
18. Assume "create table department (identifier int primary key, name varchar(50) unique not null, office varchar(30)" as well as "alter table student add column major int foreign key references department on delete restrict on update cascade", write a query that produces the count of students in each department (defined via each student's major).
    select d.identifier, d.name, count (*) as students
    from department as d, student as s
    where s.major = d.identifier
    group by d.identifier, d.name
19. For the query of the last question, write or draw an expression in relational algebra.
    … perhaps using an inner join feeding into a grouping operation
20. For the query of the last question, write or draw an alternative expression in relational algebra.
    … perhaps with the grouping on the join input (counting students per major)
21. For the query of the last question, write or draw a query execution plan.
    … perhaps using hash join and hash aggregation
22. For the query of the last question, write or draw an alternative query execution plan.
    … perhaps using group-join, or perhaps using in-sort grouping and merge join
23. Assume additional tables course (department int foreign key references department (identifier), number int, title varchar(50) not null, primary key (department, no)) and enrollment (identifier int, department int, number int, grade char(2), primary key (identifier, department, no), foreign key references student, foreign key (department, no) references course), write a query that counts the student in database courses.
    select count (*) as database_students
    from students as s join enrollment as e on (s.identifier = e.identifier)
      join course as c on (c.department = e.department and c.number = e.number)
    where c.title like "%database%"
24. Write a query that counts the number of students in each database course; produce a table with a row for course.
    select c.department, c.number, c.title, count (*) as students
    from <as above>
    where <as above>
    group by c.department, c.number, c.title
25. Same as before, but include the department names in this table.
    …
26. Write a query that finds the database course(s) with the most students; include the full department name in the output.
    …

Student questions

1. Different types of joins (left, right, inner, full, outer, and semi) - perhaps with Venn diagrams on the board to illustrate the differences?
2. Differences between the various locking implementations in ARIES: key value locking (KVL), key range locking (KRL), and index management (IM).
3. Review of log-structured merge forests, including how the trees get merged and analysis of complexity and performance.
4. Can you review how to do cardinality estimation and what specifically does robustness refer to?
5. Can you give some examples on how to do query optimization using techniques such as pushing down grouping operation, interesting ordering and count (distinct > 1)
6. The utilization of sorting in database management and some key examples where sorting is used. (external merge sort, duplicate removal..)