01:198:336 - Databases - Exam 1 Notes

Pranav Tikkawar

October 21, 2025

Contents

1 L1

Database Management System: Collection of interrelated data and a set of programs to access that data.
Online Transaction Processing: OTLP Large number of users often accessing the same data at the same time. Each user retrieves or updates small amounts of data.
Online Analytics Processing: OLAP Large amounts of data are analyzed for trends. Read heavy with low write. **Why not a file System:** File systems are not designed to handle the complexity and volume of data that databases do. Struggle with data redundancy, inconsistency, and difficulty in data retrieval.
Data Abstraction: Abstraction is the process of hiding the details and showing only the essential features of the data. The levels of data abstraction are: Physical Level, Logical Level, View Level.
Database Architecture: Naive users, data analytics applicaiton programmers, database administrators conncet with sql to the DBMS which connects to the database.
DataBase Admin: Central controller of the database. creates intial schema, how it is stored and accessed, and maintains the database.

2 L2

Relational Data Model Proposed by E.F. Codd in 1970. Data is represented in the form of tuples (rows), relations (tables), and attributes (columns). Rows are distinct and unordered. Columns ordered and named.
Relation Schema: A logical structure of the database. It defines the tables, fields, and relationships between them.
Relation Instance: A specific set of data that conforms to the structure defined by the relation schema at a particular point in time.
Keys: Let $K \subset R$ where R is a relation (table).

- **Superkey:** K are sufficent to identify a unique tuple of each possible relation instance of R.
- **Candidate Key:** A minimal superkey. No proper subset of K is a superkey.
- **Primary Key:** A chosen candidate key to identify tuples uniquely in a relation.
- **Foreign Key:** An attribute or set of attributes in one relation that refers to the primary key in another relation.

Foreign key Constraint: A forien key constratin from attributes A of a relation r_1 to the primary key B of relation r_2 states that on any instance of the database, the value of A in r_1 must either be null or must match the value of B in some tuple of r_2. Atribute A is called a forien key from r_1 to r_2. r_1 is called the referencing relation and r_2 the referenced relation.

Relational Algebra: A procedural query language that takes relations as input and produces relations as output.
Basic Operations:

- **Selection (σ):** Selects rows from a relation that satisfy a given predicate.
- **Projection (π):** Selects specific columns from a relation, eliminating duplicates.
- **Cartesian Product ($\times$):** Combines two relations to form a new relation with all possible pairs of tuples. We use Join in practice to ensure we get the tuples which follow a certain condition. we can write as $r \bowtie_{condition} s = \sigma_{condition}(r \times s)$.
- **Union ($\cup$):** Combines tuples from two relations, eliminating duplicates. They need to have the same arity (number of attributes) and corresponding attributes must have the same domain.
- **Set Intersection ($\cap$):** Returns tuples that are present in both relations. Same conditions as union.
- **Set Difference ($-$):** Returns tuples that are in the first relation but not in the second. Same conditions as union.
- **Assignment ($\leftarrow$):** Assigns the result of a relational algebra expression to a new relation. This is useful for breaking down complex queries into simpler steps.
- **Renaming (ρ):** Changes the name of a relation or its attributes. This is useful for clarity and to avoid naming conflicts in complex queries.

3 L3

SQL: Structured Query Language. A standard language for managing and manipulating relational databases.
Data Definition Language (DDL): Used to define and manage database structures. Commands include CREATE, ALTER, DROP.
Data Manipulation Language (DML): Used for querying and modifying data. Commands include SELECT, INSERT, UPDATE, DELETE.
Transaction Control: Includes commands for specifying beginning and end of transactions. Commands include COMMIT, ROLLBACK.
Embedded and Dynamic SQL: SQL statements are embedded within a host programming language like C, Java, or Python.
Basic Types: char(n) - fixed length string, varchar(n) - variable length string, int - integer, smallint - small integer, numeric(p,s) - fixed precision and scale numbers, real/double precision - floating point numbers, float - approximate floating point numbers, date - date values, time - time values, timestamp - date and time values.
Create Table: Used to create a new table in the database. Syntax: CREATE TABLE table_name (column1 datatype, column2 datatype, ..., ¡Integrity-Constraints¿);

Integrity Constraints: primary key - PRIMARY KEY (column1, column2, ...), foreign key - FOREIGN KEY (column1, column2, ...) REFERENCES other_table (other_column1, other_column2, ...), unique - UNIQUE (column1, column2, ...), not null - column_name datatype NOT NULL.
Drop Table: Used to delete a table and all its data from the database. Syntax: DROP TABLE table_name;
Alter Table: Used to modify the structure of an existing table. Syntax: ALTER TABLE table_name ADD column_name datatype; ALTER TABLE table_name DROP COLUMN column_name; ALTER TABLE table_name MODIFY COLUMN column_name datatype;
Basic Query Structure: SELECT column1, column2, ... FROM table_name WHERE condition;
Select Clause: Specifies the columns to be retrieved. Use * to select all columns. Corresponds to the projection operation in relational algebra. Can use distinct to eliminate duplicates. all is there by default which means duplicates are not removed. Can use as to rename columns in the result set.
From Clause: Specifies the table(s) from which to retrieve data. Can include multiple tables for joins. Corresponds to the Cartesian product and join operations in relational algebra. can use as to rename tables in the query: example: FROM table_name AS t1.
Where Clause: Specifies the conditions that must be met for a row to be included in the result set. Corresponds to the selection operation in relational algebra.
Rename Operator: Used to rename the output columns of a query. Syntax: SELECT column1 AS new_name1, column2 AS new_name2, ... FROM table_name as t WHERE condition; Example: SELECT DISTINCT T.name FROM instructor AS T, instructor AS S WHERE T.salary ¿ S.salary and S.dept_name = 'Physics'; Find names of all instructors who earn more than some instructor in the Physics department.
Self Join: A self join is a regular join, but the table is joined with itself. Example: SELECT A.name, B.name FROM instructor AS A, instructor AS B WHERE A.dept_name = B.dept_name AND A.salary ¿ B.salary; Find names of all instructors who earn more than some other instructor in the same department.
String Operations: The operator LIKE uses patterns that are desribed using two special wildcard characters: % and _. The % character matches any sequence of zero or more characters. The _ character matches any single character. Example: SELECT name FROM instructor WHERE name LIKE 'A%'; Find names of all instructors whose names start with 'A'.

4 L4

Ordering Results: The ORDER BY clause is used to sort the result set by one or more columns. Syntax: SELECT column1, column2, ... FROM table_name ORDER BY column1 [ASC—DESC], column2 [ASC—DESC], ...; Default is ascending order (ASC). Use DESC for descending order.
Where Clause Predicates: Comparison Operators: =, !=, ¡, ¿, ¡=, ¿=; Logical Operators:

AND, OR, NOT; BETWEEN: Checks if a value is within a range; IN: Checks if a value is in a list of values; IS NULL: Checks for null values.
Set Operations Union (UNION), Intersection (INTERSECT), Set Difference (EXCEPT). ALL keyword can be used to include duplicates.
Null Values: Represents missing or unknown data. SQL treats as unknown in comparisons. Use IS NULL and IS NOT NULL to check for null values. **Aggregate Functions**: Average (AVG), Count (COUNT), Maximum (MAX), Minimum (MIN), Sum (SUM). Used to perform calculations on a set of values and return a single value.
Aggregate Functions - Group By: The GROUP BY clause is used to group rows that have the same values in specified columns into summary rows. Syntax: SELECT column1, aggregate_function(column2) FROM table_name GROUP BY column1;
Aggregate Functions - Having: The HAVING clause is used to filter groups based on a specified condition. Predicates in the having clause are applied after the formation of groups. Syntax: SELECT column1, aggregate_function(column2) FROM table_name GROUP BY column1 HAVING condition;
Aggregate Functions - Evaluation Order: FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY.
Nested Subqueries: Is a SELECT FROM WHERE query that is embedded within another SQL query. The subquery is executed first, and its result is used by the outer query. Example: SELECT name FROM instructor WHERE dept_name IN (SELECT dept_name FROM department WHERE building = 'Science Hall'); Find names of all instructors who work in departments located in the 'Science Hall' building.
Set Membership: The IN operator is used to check if a value exists in a list of values or the result set of a subquery. Example: SELECT name FROM instructor WHERE dept_name IN ('Computer Science', 'Mathematics'); Find names of all instructors who work in the Computer Science or Mathematics departments.
Set Comparison - Some The SOME operator is used to compare a value to each value in a list or subquery result set. It returns true if the comparison is true for at least one value. Example: SELECT name FROM instructor WHERE salary ¿ SOME (SELECT salary FROM instructor WHERE dept_name = 'Physics'); Find names of all instructors whose salary is greater than at least one instructor in the Physics department.
Set Comparison - All The ALL operator is used to compare a value to all values in a list or subquery result set. It returns true if the comparison is true for all values. Example: SELECT name FROM instructor WHERE salary ¿ ALL (SELECT salary FROM instructor WHERE dept_name = 'Mathematics'); Find names of all instructors whose salary is greater than all instructors in the Mathematics department.
Exists Clause: The EXISTS operator is used to check if a subquery returns any rows. It returns true if the subquery returns at least one row. Example: SELECT name FROM department WHERE EXISTS (SELECT * FROM instructor WHERE instructor.dept_name = department.dept_name); Find names of all departments that have at least one instructor.
Test for Absence of Duplicate Tuple: To test for the absence of a duplicate tuple in a relation, we can use a nested subquery with the where unique construct. Example: SELECT T.course_id FROM as T WHERE UNIQUE (SELECT R.course_id FROM section AS R WHERE T.course_id = R.course_id AND R.year = 2017); Find course IDs of all courses

that were offered only once in the year 2017.
Subqueries in the From Clause: SQL allows a subqueries to appear in the FROM clause. EX: SELECT dept_name, avg_salary FROM (SELECT dept_name, AVG(salary) AS avg_salary FROM instructor GROUP BY dept_name) WHERE avg_salary ¿ 80000; Find names of all departments whose average instructor salary is greater than 80000.
With Clause: The WITH clause is used to define a temporary relation whose defintion is available only to the query in which the with clause occurs. Example: WITH HighSalaryInstructors AS (SELECT name, salary FROM instructor WHERE salary ¿ 90000) SELECT name FROM HighSalaryInstructors; Find names of all instructors with a salary greater than 90000 using a temporary relation.
Scalar Subquery A Scalar Subquery is one which is used where a single value is expected. Runtime error if subquery returns more than one result tuple

5 L5

DB Modifications Deletions, Insertion, Updating
Deletion Delete all instructors: DELETE FROM instructor
Insertion Insert a new instructor: INSERT INTO instructor (ID, name, dept_name, salary) VALUES (12345, 'John Doe', 'Computer Science', 75000);
Updating Update an instructor's salary: UPDATE instructor SET salary = 80000 WHERE ID = 12345;
Case Statement for Conditional Updates:Use Case statements to perform conditional updates. Example: UPDATE instructor SET salary = CASE WHEN dept_name = 'Computer Science' THEN salary * 1.1 WHEN dept_name = 'Mathematics' THEN salary * 1.05 ELSE salary END;
Joined Relations Natural Join, Inner Join, Outer Join. Takes two relations as input and combines them based on a specified condition.
Natural Join: Combines two relations based on all common attributes. Returns only one copy of the common attributes in the result. Example: SELECT * FROM instructor NATURAL JOIN department;
Dangers of Natural Join: Unrelated attributes with the same name can lead to unintended results. Use the USING clause or explicit join conditions to avoid ambiguity.
Join Condition: ON condition allows specifying the join condition explicitly. Example: SELECT * FROM instructor JOIN department ON instructor.dept_name = department.dept_name;

6 L6

Outer Join: Computes the join then adds tuples from one relation that does not match tuples in the other relation to the result of the join, filling in nulls for missing attributes.

Left Outer Join: Returns all tuples from the left relation and matching tuples from the right relation. Non-matching tuples from the right relation are filled with nulls.
Right Outer Join: Returns all tuples from the right relation and matching tuples from the left relation. Non-matching tuples from the left relation are filled with nulls.
Full Outer Join: Returns all tuples from both relations. Non-matching tuples from either relation are filled with nulls.
Joined Types and conditions: Join Operations takes two relations and return a result as another relation **Inner Join**: Returns only the tuples that have matching values in both relations.
View Definition: A view is a virtual table whose contents are defined by a query. Views are used to simplify complex queries, provide security, and encapsulate data. Example: CREATE VIEW HighSalaryInstructors AS SELECT name, salary FROM instructor WHERE salary ¿ 90000; Find names of all instructors with a salary greater than 90000 using a view.
Views Defined by other views A View relation v_1 can be defined using another view relation v_2. Depend directy: if v_1 uses v_2 in its definition. Depend indirectly: if v_1 uses a view v_3 that depends on v_2.
Materialized View: Certain DBMS support materialized views, which store the result of the view query physically. This can improve performance for complex queries but requires maintenance to keep the view updated with changes in the underlying data.
View Updates: Most SQL implimentations allow updates on simple views.
Transactions: A transaction is a sequence of one or more SQL operations treated as a single logical unit of work. Transactions ensure data integrity and consistency in the database. Have Commits and Rollbacks. Have integitry constraints.
Check Constraint: A check constraint is used to limit the values that can be placed in a column. It ensures that the data meets specific criteria before being inserted or updated. Example: CREATE TABLE employee (ID INT PRIMARY KEY, name VARCHAR(100), salary INT CHECK (salary ¿ 0)); Ensures that the salary is always greater than 0.
Referential Integrity: Ensures that foreign key values in a child table correspond to primary key values in a parent table. Prevents orphaned records and maintains consistency between related tables.
Cascading Actions: Define actions to be taken when a referenced row is updated or deleted. Options include CASCADE, SET NULL, SET DEFAULT, and NO ACTION. Example: FOREIGN KEY (dept_name) REFERENCES department(dept_name) ON DELETE CASCADE;
Assertions: An assertion is a constraint that applies to the entire database. It is used to enforce complex integrity rules that involve multiple tables. Example: CREATE ASSERTION PositiveSalary CHECK (NOT EXISTS (SELECT * FROM employee WHERE salary ¡ 0)); Ensures that no employee has a negative salary.
Large Object Types: SQL supports large object types such as BLOB (Binary Large Object) and CLOB (Character Large Object) for storing large amounts of binary or text data, respectively. These types are used for storing images, documents, multimedia files, etc.
User-Defined Types: SQL allows the creation of user-defined types (UDTs) to encapsulate complex data structures. UDTs can be used to define custom data types that suit specific application needs. Example: CREATE TYPE Address AS OBJECT (street VAR-

CHAR(100), city VARCHAR(50), zip_code VARCHAR(10));
Indexing: An index is a database object that improves the speed of data retrieval operations on a table. Indexes are created on one or more columns of a table and allow for faster searching, sorting, and filtering of data. Example: CREATE INDEX idx_dept_name ON instructor(dept_name);
Authorization and Security: SQL provides mechanisms for controlling access to database objects through user roles and permissions. Common commands include GRANT and REVOKE to manage user privileges. Example: GRANT SELECT, INSERT ON instructor TO user_name;
Role A role is a named group of related privileges that can be granted to users or other roles. Roles simplify the management of user permissions by allowing administrators to assign a set of privileges to multiple users at once. Example: CREATE ROLE read_only; GRANT SELECT ON instructor TO read_only;

7 L7-8

Transaction Concept: A unit of program exceuption that accesses and possibly modifies the contents of a database.
ACID Properties: Atomicity, Consistency, Isolation, Durability
Atomicity: The system should ensure that updates of a partially completed transaction are not reflected in the database. Either all operations of the transaction are reflected in the database or none are.
Consistency: A transaction should take the database from one consistent state to another consistent state.
Isolation: The operations of a transaction should be isolated from those of other transactions. The intermediate state of a transaction should not be visible to other transactions. This can be ensured by running transactions serially.
Durability: Once a transaction has been committed, its effects should persist in the database even in the event of a system failure.
Transaction States: Active, Partially Committed, Committed, Failed, Aborted.
Concurrent Executions: When multiple transactions are executed concurrently, the benefits are Increased processor and disk utilization, Reduced waiting time, Increased throughput.
Schedules: A schedule is a sequence of instructions that specify the chronological order in which the operations of concurrent transactions are executed.
Serializability: Each transaction is executed completely before the next transaction begins. thus this maintains the consistency of the database.
Conflicting Instructions: Two instructions are said to be conflicting if they belong to different transactions, they access the same data item, and at least one of the instructions is a write operation.
Read-Write : A read-write conflict occurs when one transaction reads a data item and another transaction writes to the same data item.
Write-Read : A write-read conflict occurs when one transaction writes to a data item and

another transaction reads from the same data item.
Write-Write : A write-write conflict occurs when two transactions write to the same data item.
Conflict Serializability: A schedule is said to be conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting instructions.
Transaction A transaction is the execution of a sequence of one or more operations on a shared database to preform some higher level function.
Precedence Graph A directed graph that helps determine whether a schedule is conflict serializable. The graph has a node for each transaction and a directed edge from transaction Ti to Tj if an operation of Ti precedes and conflicts with an operation of Tj. If the precedence graph has no cycles, the schedule is conflict serializable.
Recoverable Schedules: A schedule is recoverable if, for any pair of transactions Ti and Tj, if Tj reads a data item previously written by Ti, then the commit operation of Ti appears before the commit operation of Tj in the schedule. In other words, a transaction should not commit until all transactions from which it has read data have committed.
Cascading Rollbacks: A cascading rollback is a single transaction failure which leads to the failure of other transactions.
Cascadeless Schedules: A schedule is cascadeless if, for any pair of transactions Ti and Tj, if Tj reads a data item previously written by Ti, then the commit operation of Ti appears before the read operation of Tj in the schedule. In other words, a transaction should only read data that has been committed by other transactions. This also means that cascadeless schedules are recoverable and do not suffer from cascading rollbacks.

8 L9

Concurrency Control: Mechanism that will ensure that all possible schedules are conflict serializable.
Transactions in SQL: BEGIN; TRANSACTION COMMIT: - makes all changes made in the transaction permanent. ABORT or ROLLBACK: - undoes all changes made in the transaction. AUTOCOMMIT mode: each individual SQL statement is treated as a transaction and is automatically committed right after it is executed.
Mechanisms for Isolation: Concurrency control protocol is how the DBMS decides the proper interleaving of operations from multiple transactions.
Isolation Anomalies: Ditry Read - A transaction reads data written by a uncommitted transaction. Unrepeatable Read - A transaction reads the same data item twice and gets different values each time because another transaction has modified the data item in between the two reads. Phantom Read - A transaction re-executes a query that retrieves a set of rows that satisfy a certain condition and finds that the set of rows has changed because another transaction has inserted or deleted rows that satisfy the condition.
Implimentation of Isolation Levels: Lock-Based Protocols, Timestamp-Based Protocols, Validation-Based Protocols. Two major categories of concurrency control protocols are pessimistic and optimistic.

Basic Lock Types: Shared Lock (S-Lock) - shared lock for reads, Exclusive Lock (X-Lock) - exclusive lock for writes
Executing With Locks Transaction must request locks, Lock manager grants or blocks requests, Transaction releases locks.
2Phase Locking (2PL): A concurrency control protocol that ensures serializability by requiring that all locking operations precede the first unlock operation in a transaction. Sufficient to guarantee conflict serializability. Subject to cascading aborts and unrecoverable schedules.
2 Phases: Growing Phase - A transaction may obtain locks but may not release any locks. Shrinking Phase - A transaction may release locks but may not obtain any new locks.
Lock Conversions Growing Phase: Any acquiring of a lock must be done before any releasing of a lock, and upgrading a lock from S to X is allowed. Shrinking Phase: Downgrading a lock from X to S is allowed, but no new locks can be acquired.
Strict 2PL: A special case of 2PL where a transaction holds all its exclusive locks until it commits or aborts. Ensures cascadeless schedules and recoverability.
Rigorous 2PL: A special case of 2PL where a transaction holds all its locks (both shared and exclusive) until it commits or aborts. Ensures serializability, cascadeless schedules, and recoverability.
Deadlocks: A situation where two or more transactions are waiting indefinitely for each other to release locks.
Deadlock Detection: DBMS makes a waits-for graph to keep track of transactions waiting for locks. If a cycle is detected in the graph, a deadlock exists.
Deadlock Handling: DBMS will select a victim transaction to abort and release its locks, allowing other transactions to proceed. The victim is typically chosen based on factors such as the amount of work done, the number of resources held, or priority levels.
Victim Selection: DMBS chooses by age, prohress, number of tiems locked, number of transaction needed to rollback. For rollback length 2 approaces for completely rollback or minimal rollback.
DeadLock Prevention: When a transaction requests a lock that cannot be granted immediately, the DBMS can choose to abort the transaction instead of making it wait. This approach prevents deadlocks by ensuring that transactions do not hold locks while waiting for other locks. Assign priority based on timestamps. Wait-Die: Older transaction waits for younger one, Younger transaction aborts if it requests a lock held by an older one. Wound-Wait: Older transaction aborts younger one if it requests a lock held by the older one, Younger transaction waits for older one.

9 L10

Storage Hierarchy: Registers, Cache, Main Memory, Secondary Storage, Tertiary Storage. Primary storage - Fast, volatile, expensive. Secondary storage - Slow, non-volatile, cheap, Tertiary storage - very slow, non-volatile, very cheap.
Magnetic Tapes: Very slow, 300mb/sec. Cheap, long term storage. Sequential access.

Magnetic Disks: Faster than tapes, Still slow, 10000-150000 IOPS. Random access.
RAID: Redundant Array of Independent Disks. Improves performance and provides fault tolerance by distributing data across multiple disks.
SSD: Solid State Drive. Faster than magnetic disks, 100000-500000 IOPS. More expensive per GB. No moving parts.
Main Memory: RAM. Fast access, volatile storage. Used for temporary data storage during processing.
Cache : Small, fast memory located close to the CPU. Stores frequently accessed data to reduce latency.
Storage Manager: The storage manager is a component of the DBMS that is responsible for managing the storage of data on disk. It handles tasks such as data organization, indexing, and retrieval.
Database Pages: Page is a fixed size block of data that is the basic unit of data transfer between disk and main memory. Typical page sizes are 4KB, 8KB, or 16KB.
Fixed Length Records: Records with a fixed size. Easier to manage and access, but can lead to wasted space if records are smaller than the allocated size.
Variable Length Records: Records with a variable size. More flexible and efficient in terms of space utilization, but can be more complex to manage and access.
Slotted Page Structure: A page structure that uses a slot directory to manage variable-length records. The slot directory contains pointers to the actual records on the page, allowing for efficient access and management of variable-length records.
Large Objects: BLOBs (Binary Large Objects) and CLOBs (Character Large Objects) or stoed as files in file system or files managed by the DBMS.
Organization of Pages Heap file is an unordered collection of pages where recorded that are inserted in any order.
Linked List Heap File: Each page contains a pointer to the next page in the file. New pages are added at the end of the list.
Page Directory Heap File: A directory page is used to keep track of all the pages in the heap file. Each entry in the directory contains a pointer to a data page and information about the amount of free space on that page.
OLTP and OLAP: OLTP systems are optimized for transaction processing, while OLAP systems are optimized for data analysis and reporting. OLTP systems typically use normalized database schemas to minimize data redundancy and ensure data integrity, while OLAP systems often use denormalized schemas (such as star or snowflake schemas) to improve query performance.
Row Store vs Column Store: Row Store databases store data in rows, making them well-suited for OLTP workloads where transactions involve multiple columns of a single row. Column Store databases store data in columns, making them ideal for OLAP workloads where queries often involve aggregating data across many rows but only a few columns.

10 L11

Buffer Manager: Manage memory and move data between disk and main memory.
Buffer Pool: Large range of memory allocted on Boot as an arry of fized size pages.
Dirty and Pinned Frames: Dirty - modified pages in buffer pool that have not been written to disk. Pinned - pages that are currently being used by a transaction and cannot be replaced.
Buffer Replacemnt Policies: Goals - Correctness, Accuracy, Speed, Meta-data Overhead.
LRU: Least Recently Used. Replaces the page that has not been used for the longest time.
Clock Replacement: approximately LRU. Each page has a reference bit. When a page is accessed, its reference bit is set to 1. When a page needs to be replaced, the clock hand scans the pages. If the reference bit is 0, the page is replaced. If the reference bit is 1, it is set to 0 and the clock hand moves to the next page.
Issues with LRU/Clock: Susceptible to sequential flooding. A large sequential scan can evict all pages from the buffer pool, leading to poor performance for other transactions.
LRU-K: Tracks the K-th most recent access time for each page. When a page needs to be replaced, the page with the oldest K-th access time is selected for replacement. This approach helps to mitigate the effects of sequential flooding by considering multiple recent accesses rather than just the most recent one.
Most Recently Used (MRU): Replaces the page that was most recently used. This policy is based on the observation that in some workloads, recently used pages are less likely to be used again soon.
LFU: Least Frequently Used. Replaces the page that has been used the least number of times. This policy is based on the idea that pages that are accessed frequently are more likely to be needed again in the future.

11 Practice Midterm:

11.1 Buffer Pool Replacement:

3 Frames, Reference String: 1,2,3,1,4,2,5,1,2,3,4,2

- LRU:

Current	Hits	Miss	Content
1	0	1	(1,-,-)
2	0	1	(1,2,-)
3	0	1	(1,2,3)
1	1	0	(1,2,3)
4	0	1	(1,4,3)
2	0	1	(1,4,2)
5	0	1	(5,4,2)
1	0	1	(5,1,2)
2	1	0	(5,1,2)
3	0	1	(3,1,2)
4	0	1	(3,4,2)
2	1	0	(3,4,2)
total	3	9	(3,4,2)

Table 1: LRU Replacement

- MRU:

Current	Hits	Miss	Content
1	0	1	(1,-,-)
2	0	1	(1,2,-)
3	0	1	(1,2,3)
1	1	0	(1,2,3)
4	0	1	(4,2,3)
2	1	0	(4,2,3)
5	0	1	(4,5,3)
1	0	1	(4,1,3)
2	0	1	(4,2,3)
3	1	0	(4,2,3)
4	1	0	(4,2,3)
2	1	0	(4,2,3)
total	5	7	(4,2,3)

Table 2: MRU Replacement

- LFU (with ties broken by LRU):

Current	Hits	Miss	Content
1	0	1	(1,-,-)
2	0	1	(1,2,-)
3	0	1	(1,2,3)
1	1	0	(1,2,3)
4	0	1	(1,4,3)
2	0	1	(1,4,2)
5	0	1	(1,5,2)
1	1	0	(1,5,2)
2	1	0	(1,5,2)
3	0	1	(1,3,2)
4	0	1	(1,4,2)
2	1	0	(1,4,2)
total	4	8	(1,4,2)

Table 3: LFU Replacement

- Clock

Current	Hits	Miss	Content	Ref Bits
1	0	1	(1,-,-)	(1,0,0)
2	0	1	(1,2,-)	(1,1,0)
3	0	1	(1,2,3)	(1,1,1)
1	1	0	(1,2,3)	(1,1,1)
4	0	1	(4,2,3)	(1,0,0)
2	1	0	(4,2,3)	(1,1,0)
5	0	1	(4,2,5)	(1,0,1)
1	0	1	(4,1,5)	(0,1,1)
2	0	1	(2,1,5)	(1,1,0)
3	0	1	(2,1,3)	(1,0,1)
4	0	1	(2,4,3)	(0,1,1)
2	1	0	(2,4,3)	(1,1,1)
total	3	9	(2,4,3)	(1,1,1)

Table 4: Clock Replacement

12 Things to Review

Conflict Serialiability and Safety Properties