Midterm review

Kathleen Durant PhD

CS 3200 Northeastern University

Outline for today

- Identify topics for the midterm
- Discuss format of the midterm
  - What will be provided for you and what you can bring (and not bring)
- Review content

Midterm

- Monday in class
- Open books and open notes
  - But no portable devices (no laptops, no phones, etc.)
- Create an exam that takes on average 60 minutes to complete
  - With a standard deviation of 15 minutes we should have most students finishing the exam
  - The example midterm is longer than expected
    - It provides example of all types of questions

Text chapters for the midterm

- Chapters 1-5
    1. Overview of databases
    2. Introduction to database design ERD
    3. The relational model
    4. Relational algebra and relational calculus
    5. SQL queries – Simple queries, multiple tables, JOIN, Nested queries
- Overview of Chapter 20
    - Functional Dependency and Normal Form
- Use Safari book for My SQL’s SQL dialect

Topics for the midterm

- Modeling
  - Relational Model
  - ERD diagrams
- Query representation
  - Relational algebra
  - Relational calculus
- SQL as a Query Language
  - ALL SQL commands
  - Database constraints
- My SQL data types and system defined functions
- Tiered architectures
- User session variables

Format of the midterm

- 3-4 open-ended responses (75%)
  - You define an ERD diagram given a verbal description
  - You define the SQL code given a verbal description or you describe the result set given a dataset and a query
  - Relational calculus and relational algebra
    - You provide the expression or interpret an expression
- Some close-ended responses (25%)
  - Short collection of True and False
  - Multiple choice
  - Short definitions

How to study

- Go over the lecture notes and queries
- Read the book
  - Summary section of the chapters are written well
- Go over homework 1 & 2 & 3
  - Midterm questions based on content from 1, 2, 3 will not be as difficult as the homework problems
- Do problem sets 4 & 5
- Ask questions in piazza or via email
- Organize a study sheet
- Complete the example mid-term
- Practice problems

Modeling

- ERD
  - Entity sets. Attributes and relationships
  - Constraints: primary keys, foreign key (referential constraint) and cardinality
  - Weak entity set, ISA
- Convert ERD to relational model (create table)
  - Use the cardinality ratios and the primary key to help you go from entities and relations to tables
  - Remember if you have m-to-m you need a mapping table
  - Links are implemented via foreign keys
  - I will provide the basic shapes and arrows – you can override what is provided as long as you document
- Functional Dependency between 2 fields in a table

Example ERD Question

- Create an ER model from the following facts… X has a Y. There is always 1 U for every Z …
- Transform your diagram into a set of relations.

Relational Algebra

- Key points
  - Know the fundamental operators and join
  - Problems will not go beyond 2 or 3-way JOIN
  - I will provide the legend of operations for you
  - Selection (σ): Selects a subset of tuples from a relation.
  - Projection (π): Selects columns from a relation.
  - Cross-product ($\times$): Allows us to combine all tuples on 1 relation to all in other relation.
  - Set-difference ($-$): Tuples in relation 1, but not in relation 2.
  - Union ($\cup$): Tuples in relation 1 and in relation 2.
  - Join ($\bowtie$): Join tables using a conditional
  - Examples of relational algebra expressions will be provided on test

RA: Operator composition example.

Select the sailor id and name of sailors with id > 55516

Select and Project

S1

SID	Name	Login	DoB	GPA
55515	Smith	smith@ccs	Jan 10,1990	3.82
55516	Jones	jones@hist	Feb 11, 1992	2.98
55517	Ali	ali@math	Sep 22, 1989	3.11
55518	Smith	smith@math	Nov 30, 1991	3.32

$$\pi_{Sid,\ Name}\,(\,\sigma_{Sid\ >\ 55516}\,(\mathrm{S1}))\quad \mathrm{OR}\quad \sigma_{Sid\ >\ 55516}(\pi_{Sid,\ Name}(\mathrm{S1}))$$

SID	Name
55517	Ali
55518	Smith

Domain Relational Calculus

- Key points
  - You are creating a set that satisfies a condition
  - Domain variables in the bind clause determined by the order of the columns in the tuple
- Examples
- **Query 1:** Find all employees whose salary is greater than 30,000
- $\{ < id,n,b,a,s,d > \mid < id,n,b,a,s,d > \in EMPLOYEE \land s > 30{,}000 \}$

- **Query 2:** Find the employees records who works in department number 1
- $\{ < id,n,b,a,s,d > \mid (< id,n,b,a,s,d > \in EMPLOYEE \land d = 1) \}$

1ST Normal Form

- No repeating entities or group of elements
  - Do not have multiple columns representing the same type of entity
  - Primary key that represents the entity

2^{nd} Normal Form

ONLY APPLIES TO COMPOSITE KEYS

- Schema must be in first normal form
  - You have eliminated group sets
  - Every tuple has a unique key
- Each field not in the primary key provides a fact about the entity represented via the (entire) primary key
  - The primary key must be minimal – no extra fields thrown in
  - No partial dependency on part of the primary key
    - Only applies to composite primary key
- Helps you identify a relation that may represent more than one entity
- All fields must be functionally dependent on the complete primary key

3rd Normal Form

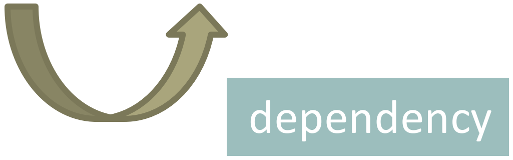
- No functional dependency between 2 non-key attributes
- Typically the form most database developers strive to be at

- Bill Kent: Every non-key attribute must provide a fact about the key, the whole key and nothing but the key

Example: Functional Dependency

- Social Security #, Name, Lot, Rating, Wage, Hours per week

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40


dependency

- FDS S → {S,N,L,R,W,H} AND R → W
- Problems due to R → W :
  - *Update anomaly*: Can we change W in just the 1st tuple of SNLRWH?
  - *Insertion anomaly*: What if we want to insert an employee and don't know the hourly wage for his rating?
  - *Deletion anomaly*: If we delete all employees with rating 5, we lose the information about the wage for rating 5

SQL

- Be able to create a SELECT statement from an English statement, relational algebra or relational calculus
  - Be proficient with using Group By, Intersection, Union, Inner Join, IN, aggregator functions such as count, sum, min, max
- Be able to provide a complete specification of CREATE TABLE
  - Constraints (Check, Foreign Key)
- Understand sub-queries
- Know the functionality of the other SQL commands
  - Be able to interpret the results of all SQL commands

Data Definition Commands

- Create a database , table
- Alter a table
  - Add a column, constraint, index
  - Alter a column by adding/deleting defaults
  - Modify a column – name, type
  - Change a column (change name), column ordering
  - Drop a column, primary key, index, foreign key
- Drop a database
- Create a table
- Drop a table
- Alter table

SQL CREATE

- Allows you to define the name of the table, name of the fields, the type of each field, and any constraints on the data
- **CREATE TABLE <table name> (<col-def-1>,**
- **<col-def-2>, … <col-def-n>,**
- **<constraint-1>, … <constraint-k>)**
  - You need a declaration for each field in the table even if it is a foreign key
  - If you do not provide a referencing field for a foreign key constraint the primary key of the table is assumed
    - This does not work in My SQL, you must always provide the field name

SQL CREATE

- Types of constraints
  - **PRIMARY KEY (specify a primary key)**
  - **UNIQUE (specify a field must have unique values)**
  - **FOREIGN KEY (specify a reference to another table)**
  - **INDEX (specify a field to be indexed) (not studied yet)**
  - **CHECK – limit the values of a field**

-
```
Create table treats(patientid int, nurseid int,
        primary key (patientid, nurseid),
        foreign key  (patientid) references patient(patientid)
                    on delete cascade,
        foreign key(nurseid) references nurse(nurseid)
                    on delete cascade,
         check patientid > 0  ) ;
```

Creating constraints after Create time

- ALTER TABLE Patients ADD CHECK (Pid>0)

Data Manipulation Language

- SELECT – retrieves data from a database
- INSERT – insert new records into a table
- UPDATE – Update a set of records from an existing table
- DELETE – Deletes a set of records from a table
- REPLACE – INSERT record into the DB. If a record exists with the same primary key the new record replaces the old record (MY SQL specific – only makes sense for tables with primary keys)

SELECT syntax

- SELECT DISTINCT *expression [,expression]* FROM
  table-expression WHERE *where-expression*
  [GROUP BY {*col_name* | *expr* | *position*}
  [ASC | DESC]]
  [HAVING *where_condition*]
  [ORDER BY {*col_name* | *expr* | *position*} [ASC | DESC], ...]
  [LIMIT {[*offset*,] *row_count* | *row_count* OFFSET *offset*}]

Expression is a field name or an expression
Table-expression is a table name
table1 [LEFT|RIGHT] [INNER|OUTER] JOIN table2
ON table1.keyvalue = table2.keyvalue
table name [CROSS] JOIN table name
table name, table name
Where condition evaluates to true for each row to be selected

SET Operators in My SQL

- UNION - join similarly shaped sizes together
  - Tables must have the same number of columns and the corresponding columns must have same data type
  - Example select name from t1 union select name from t2;
- SET DIFFERENCE
  - SELECT * FROM a WHERE (x,y) NOT IN (SELECT * FROM b);
- INTERSECTION
  - SELECT * FROM a WHERE (x,y) IN (SELECT * FROM b)

JOINS

- INNER JOIN: This will only return rows when there is at least one row in both tables that match the join condition.
  - Equi-join : performs an INNER JOIN using the equality operator
    - Most common JOIN using foreign keys
  - NATURAL JOIN: Equi-join performs an INNER JOIN using equality based on the fields that have common names between the two tables
    - Should typically not be used since dependent on tables' schemas
    - Returns pairs of rows with common values for identically named columns and without duplicating columns
  - returns pairs of rows satisfying a condition
- OUTER JOIN
  - LEFT OUTER JOIN (or LEFT JOIN): This will return rows that have data in the left table (left of the JOIN keyword), even if there's no matching rows in the right table.
  - RIGHT OUTER JOIN (or RIGHT JOIN): This will return rows that have data in the right table (right of the JOIN keyword), even if there's no matching rows in the left table.
  - FULL OUTER JOIN (or FULL JOIN): This will return all rows, as long as there's matching data in one of the tables. (FULL not in My SQL)
- CROSS JOIN: A CROSS PRODUCT OF BOTH TABLES

INNER JOIN: BOTH TABLES CONTRIBUTE

- Want to match records from one table with a record from the other table based on a criterion
  - If the criterion is not TRUE no record in the result set
- Typical JOIN performed by most queries is an INNER JOIN based on equality
- SELECT <field-list> from T1 **JOIN** T2 **ON** **T1.FIELD = T2.field**

Outer JOIN

- Sometimes you want to define the number of records in your resultant set
  - Do not want to drop records just because there is not a corresponding record in another table
    - One table is Optional and another table is Mandatory
    - Foreign key
  - Outer JOIN construct allows you to set it to the number of records in one or both of the tables
  - New functionality not provided via an implicit Join
- Three different variations
  - LEFT OUTER JOIN – table to the left of the JOIN decides resultant set
  - RIGHT OUTER JOIN – table to the right of the JOIN decides resultant set
  - CROSS JOIN – one record in result set for each record in the left and the right
    - Returns all pairs of rows from A and B
    - Number of records between MAX(count(A), count(B)) to (M+N)

FULL OUTER JOIN OR CROSS JOIN EXAMPLE

Select Sailors and the boats they have reserved – want to see sailors who have never made a reservation as well as reservations not assigned to a Sailor

CROSS JOIN in My SQL , FULL OUTER JOIN Oracle,

```
SELECT S.sname from sailors S CROSS JOIN Reserves R
          ON S.sid = R.sid
```

NULL Values

- When there are no NULLs in the data, conditions evaluate to true or false, but if a null is present, a condition will evaluate to the third value (‘undefined’, or ‘unknown’).
- In SQL only the tuples where the condition evaluates to true are returned.
  - False and unknown not returned
- Arithmetic operations applied to NULL - the result is NULL

- Separate function to test for NULL
  - Example: SELECT Name from Sailor where rating IS NOT NULL or the contrary (IS NULL)

3-VALUED LOGIC

X	Y	X AND Y	X OR Y	NOT X
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	UNKNOWN	UNKNOWN	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
UNKNOWN	TRUE	UNKNOWN	TRUE	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN	UNKOWN	UNKNOWN
UNKNOWN	FALSE	FALSE	UNKNOWN	UNKNOWN
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	UKNOWN	FALSE	UNKNOWN	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

FALSE = 0, TRUE = 1, UNKNOWN =1/2 NOT(X) = 1-X, AND(X,Y) =MIN(X,Y), OR(X,Y) = MAX(X,Y)

SQL: NULLs in conditions

- Select SID from Sailor where rating > 5
- Execution: rating > 5 evaluates to ‘unknown’ on the last tuple

SID	Sname	Rating	Age
28	Yuppy	9	35.0
31	Lubber	3	55.5
44	Guppy	5	35.0
58	Rusty	NULL	35.0

SID	Sname	Rating	Age
28	Yuppy	9	35.0

SQL with NULLS: Aggregates

- Select avg(Rating) as AVG, COUNT(Rating) as NUM, COUNT(*) as ALLNUM, SUM(Salary) as SUM from Sailors

  - AVG = 5.67
  - NUM = 3
  - ALLNUM = 4
  - SUM = 17

SID	Sname	Rating	Age
28	Yuppy	9	35.0
31	Lubber	3	55.5
44	Guppy	5	35.0
58	Rusty	NULL	35.0

User Session Objects

- Literals
  - Text single quoted strings
  - Numbers
- Database objects: databases, tables, fields, procedures and functions
  - Can set a default database/schema
  - Can also provide full context: *database.table.field*
  - System defined functions
  - User defined functions and procedures
- User session Variables
  - Allows you to store the results of a query
  - Can be passed to a function or another query

User Session Variables

- Variable are accessible by using the @ operation
  - @variablename
- Use set to assign a value to a variable
  - Set @var = 1; or @var := 1;
  - Variables not assigned a value has a default value of NULL
  - Can use other methods to assign a variable value
- Data type for a variable is determined by the last assigned value
- User variables can be assigned a value from a limited set of data types: integer, decimal, floating-point, binary or non-binary string, or NULL value
  - Assignment of other types are converted to one of the above listed types

Session variables

- Variables referenced using the @ sign
- S contains the sets of sailor record ids
- @s = select id from Sailors;

- Can be used to pass value into a function
- @s := ‘a’;
- @a = concat(“select id from Sailors where name”,@s)

My SQL Functions

- No the classes of functions
- Be able to apply a My SQL system function given a definition of it
- There are a lot I do not expect you to know them all by name but you should be able to apply the functions that were shown in class
  - .SQL files

Tiered Architecture

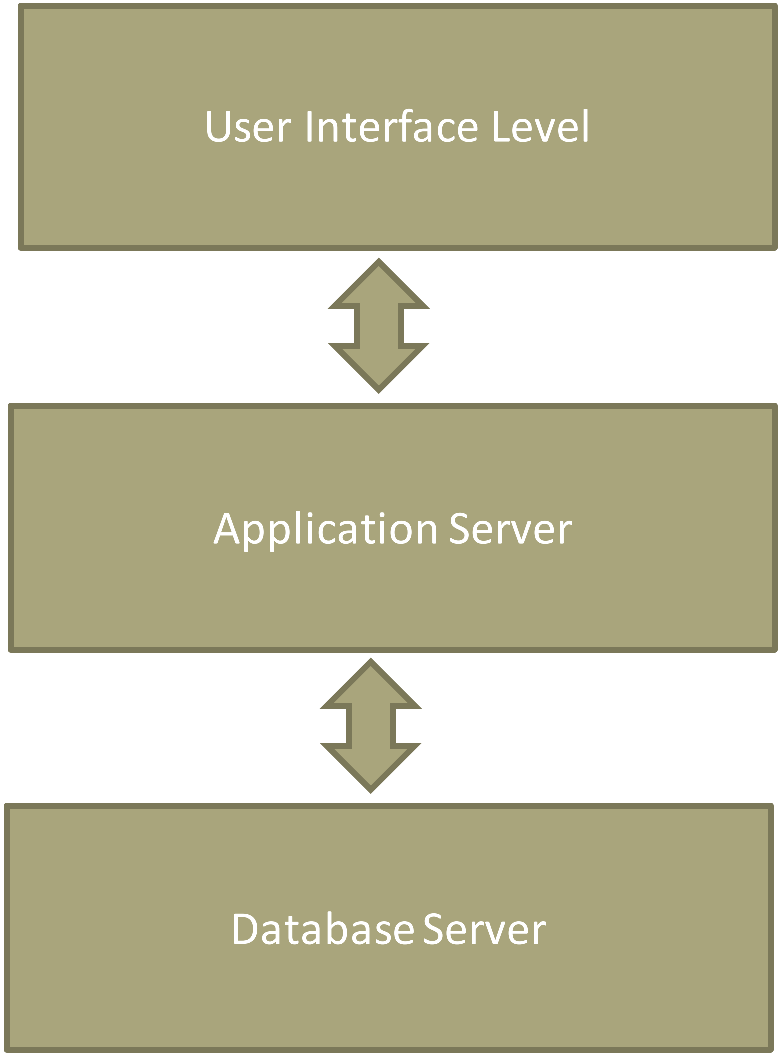
- 1, 2, 3, N level architecture
- Identifying an architecture given a system description
- Describe the benefits of the different configurations

3 Tier Architecture

FIRST TIER TASKS
User interface

SECOND TIER TASKS
Business Logic
Data processing logic

THIRD TIER TASKS
Data validation
Database access



User Interface Level

Application Server

Database Server

3 Tiered Architecture

DESCRIPTION

- Client is only responsible for the application’s user interface
  - Simple error checking on input data
  - Leads to a ‘thin client’
- Core business logic of the application resides in its own layer
  - One application server is designed to serve multiple clients
- Addresses the problem of enterprise scalability

BENEFITS

- Less expensive hardware for the clients because client proccesses are thin
- Application maintenance is centralized
- Tier hardware configurations are independent from each other
- Load balancing is easier due to separation of the core business logic from the database functions

That’s it

- Go over the lecture notes
- Read the book
- Review the SQL files on blackboard
- Go over homework 1 , 2 , 3
    - Midterm questions will not be as difficult as homework problems
- Do homework assignment 4, 5
    - 5 to practice writing SQL code
- Ask questions in piazza or via email
- Organize a study sheet
- Complete the example mid-term
- Practice problems