DDL / DML / SQLLite / MySQL

Lecture 4

Monday - Sep 11, 2023

Housekeeping

- Homework 3 due Tuesday (tomorrow), 9/12
- Project deliverable 3 (video) due Friday.
- Check your grades to verify that we've got it all correct!
- Verify your group assignments, needed for deliverable 3!

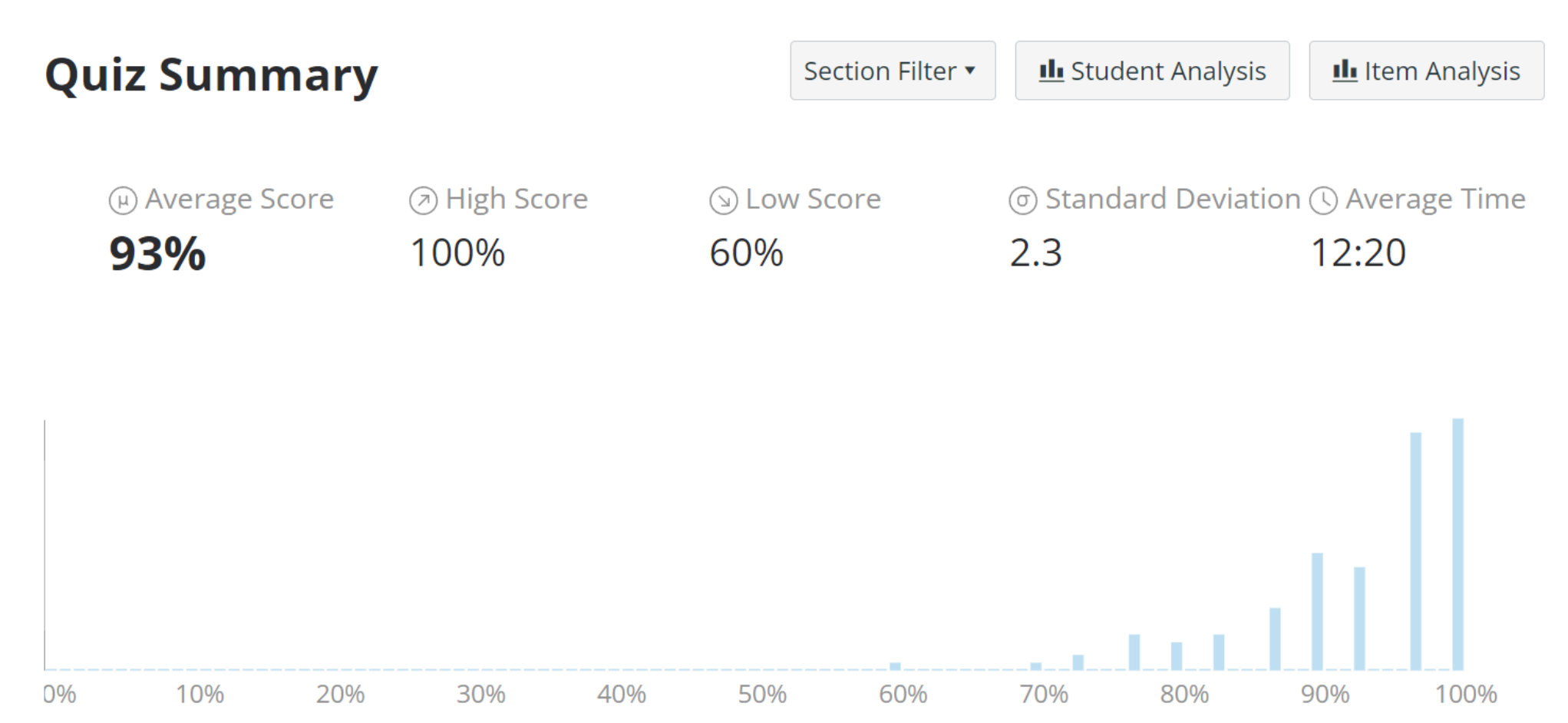
Module	Week	Date	Day	Lectures/Quizzes	Deliverables/Notes
Relational Alg.	4	9/11	Mon	MTG5: L4 (DDL / DML / SQLLite / MySQL)	
Relational Alg.	4	9/12	Tue		HW3 due (ER Models 2 - extended)
Relational Alg.	4	9/13	Wed	MTG6: L5 (Relational models)	
Relational Alg.	4	9/15	Fri		PrjDel 3 due (Topic proposal video)
Relational Alg.	5	9/18	Mon	MTG7: L6 (Relational Algebra 1)	
Relational Alg.	5	9/20	Wed	MTG8: L7 (Relational Algebra 2)	
Relational Alg.	5	9/24	Sun		HW4 due (Relational Algebra Study Guide)
Normal forms	6	9/25	Mon	MTG9: Quiz 2 today (Relational Algebra)	

Quick review of deliverables

Project deliverable 3 - video proposal

- Easiest may be to record in ZOOM
- LOG IN to zoom using SSO and your VCU account
- If your video isn’t saved, then manually move it to Kaltura
- Submit the Kaltura URL to Canvas.
- I’m experimenting with peer review. You’ll get to see and comment on videos from other students.

Quiz results and commentary



https://virginiacommonwealth.instructure.com/courses/85746/quizzes/174462/statistics

Homework 3 - ER Diagrams part 2

- Deadline extended to tomorrow (Tuesday, 9/12) night.
- General workflow:
    1. create your copy of assignment repo
    2. clone repo to local machine
    3. begin to develop solutions
    4. render document and view it in your browser.
    5. commit iterim results to GITHUB
    6. iterate steps 3-6 until done.
- SUBMIT the rendered HTML file as an attachment to Canvas.

Catching our collective breaths and taking stock

Course to date

Topics

- Database design
- Conceptual design
  - entities, attributes and relationships
  - Chen diagrams
- Logical design
  - Cardinality and Participation
  - Crow's foot diagrams

Skills and tools

- Quarto workflow and output formats
- GITHUB and git
- graphviz (not just for ER diagrams!)
- mermaid (not just for ER diagrams!)
- Zoom videos and Kaltura

Next up:

- Physical design
  - Bridging the gap between design and implementation
  - Converting multivalued attributes and relations to entities
  - Converting entities to tables and connecting them up

Databases and Database Management Systems

A database ecosystem

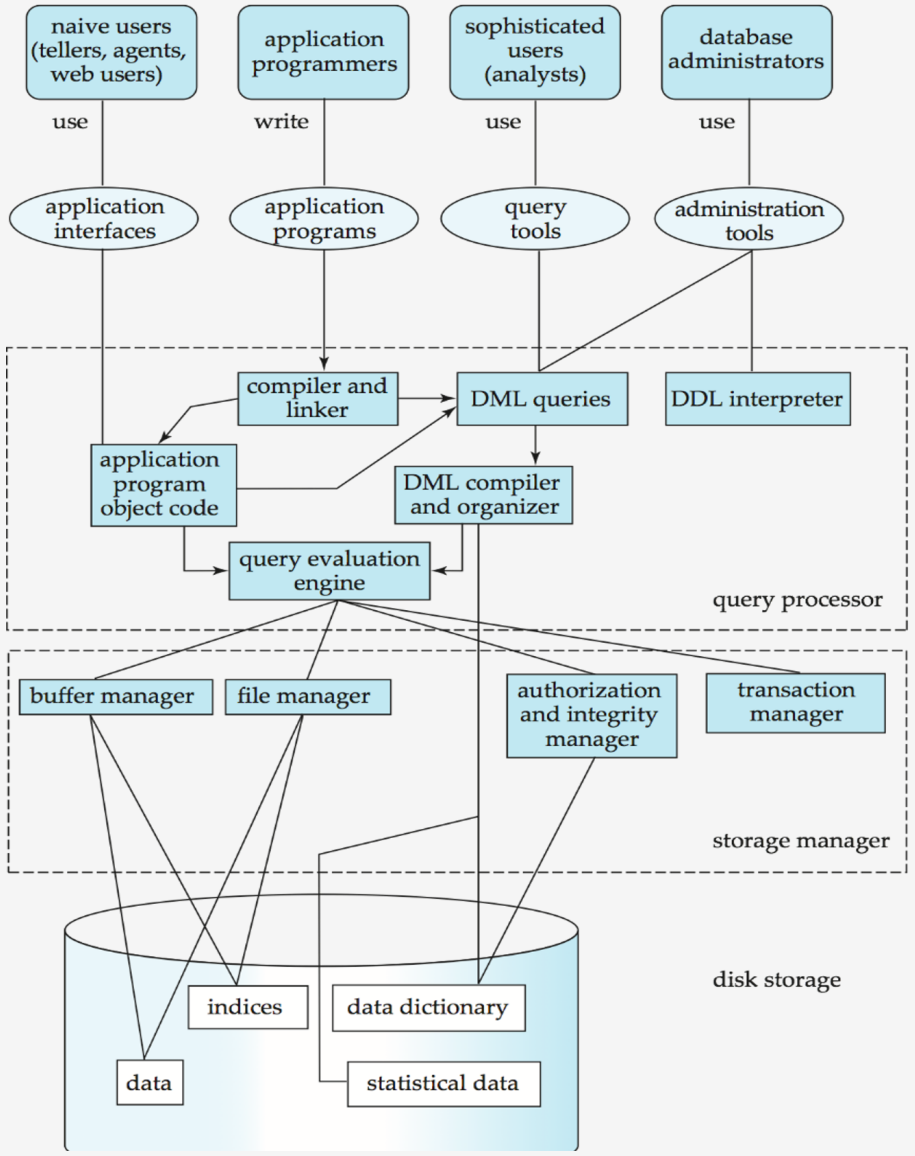
A heck of a lot more than just data!

- Users and their needs
- Ways to connect
- Query processor
- Storage manager
- Physical storage

Designing a database

Good design requires that we develop a better understanding of the database itself, so that we can more smoothly translate the conceptual and logical designs into a physical design.

The choice of database and DBMS may constrain or shape our final design.



Users and Uses

End user
- Uses application interfaces to access database

Application programmer
- Develops application interfaces (e.g., Web, API, mobile)

Business Analyst
- Uses query tools to directly access the data base (e.g. statistical business analyst)

Database Administrator (DBA)
- Designs logical and physical schemas
- Handles security and authorization
- Oversees data availability, crash recovery, software updates
- Monitors performance and tunes database as needs evolve
- Manages users and access control

Storage manager

Storage manager

- a program that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

Responsible for

- Interaction with the file manager
- Efficient storing, retrieving and updating of data

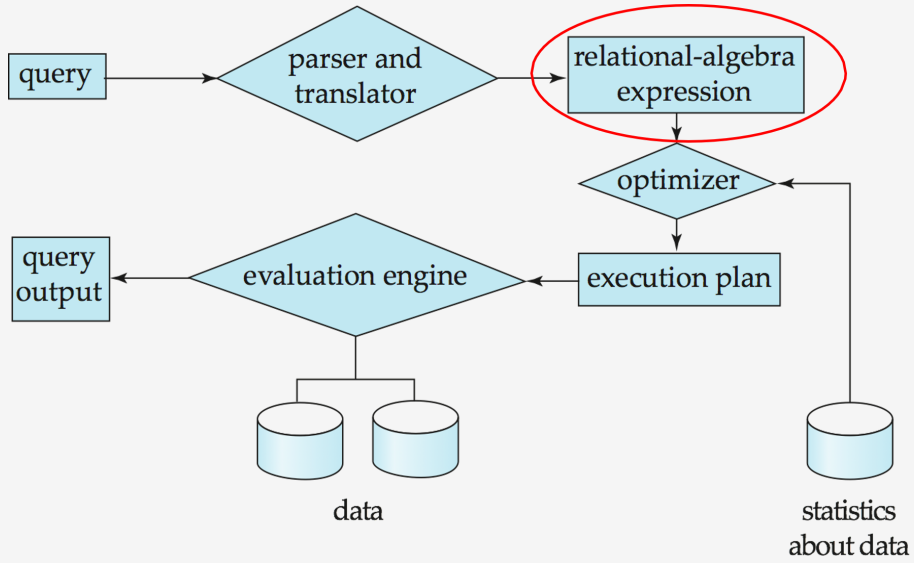
Considerations

- Types of storage access
- Types of file organization
- Types of indexing and hashing
- Types of authorization and access control
- Data integritity

Query manager

Lots going on in between request and response!

- REQUEST (query)
  - query parser
  - relational algebraic evaluationr
  - optimizer
  - planner (detailed code)
  - evaluator
- RESPONSE (query output)

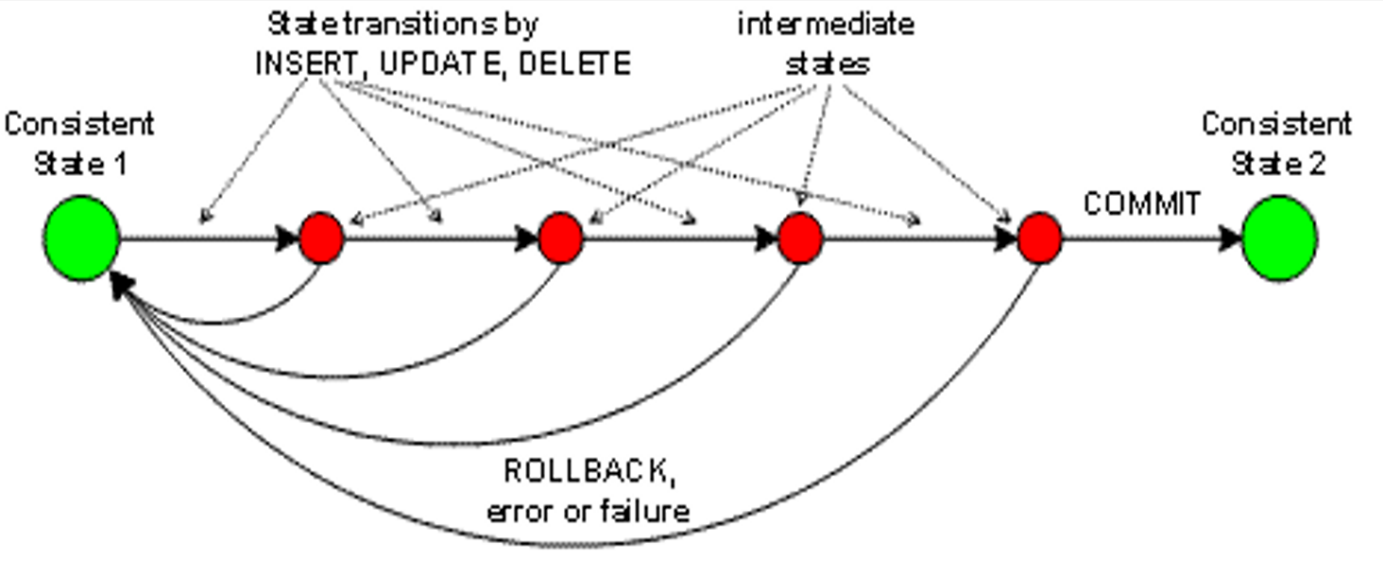


Transaction example

A **transaction** is a collection of operations that performs a single logical function in a database application

Transaction-manager ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.



ACID properties

Atomicity

Either all operations of the transaction are properly reflected in the database or none are

Consistency

Execution of a transaction in isolation preserves the consistency of the database

Isolation

Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. *For every pair of transactions Ti and Tj, it appears to Ti that either Tj, finished execution before Ti started, or Tj started execution after Ti finished*

Durability

After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures

Data Definition Language (DDL)

- DDL compiler generates a set of tables stored in a data dictionary
- Data dictionary contains metadata about:
  - Database tables, fields and data types
  - Primary and foreign keys
  - Data integrity constrants
  - Referential integrity
  - Access and authorization rules

Example

```
CREATE TABLE person (
    person_id INT PRIMARY KEY,
    person_first_name VARCHAR(50),
    person_last_name VARCHAR(50),
    person_city VARCHAR(50)
);
```

Notes

1. Fields are strongly typed,
2. INT, VARCHAR(width), (lots more!)
3. Additional attributed can be defined:
   - PRIMARY KEY, FOREIGN KEY, etc.

SQL (Structured Query Language) is the most widely used query declarative language for accessing relational databases.

Data Manipulation Language (DML)

Provides statements for all the other non-data related activities in the data base:

- managing users,
- tuning storage,
- backing up and replicating the data base,
- managing physical storage on the system,
- communicating with other systems.

SQL (Structured Query Language) is the most widely used query declarative language for accessing relational databases.

Accessing the DBMS

Using a command line

SSH into server to script creation and population of the databases.

Using a *connector*

Using an API

ODBC, REST, graphQL or other approaches to manipulate aspects of the data base.

Using PHPMyAdmin

Usually through a web browser to manage, monitor and tune the overall system.

Using a PHP full stack

To run the application and use the database.

- LAMP Linux : Apache : MySQL : PHP/Perl/Python

Using NodeJS full stack

To run the application and use the data base

- MEAN Mongo/MySQL : Express : Angular : Node
- MERN Mondo/MySQL : Express : React : Node

DBMS used in CMSC 508

MySQL

MySQL is a prominent open-source relational database management system that operates on a client-server model, making it a great choice for handling large-scale database applications and supporting multi-user environments.

Deployment and Setup
: Requires a separate server setup, involving a more complex installation process and server maintenance.

Concurrent Access
: Can handle multiple users and applications simultaneously, making it a better fit for web applications or systems with high concurrency requirements.

Storage and Performance
: Often better suited for large datasets and offers higher performance for complex queries, thanks to optimization features and more expansive indexing options.

SQLLite

SQLite is a self-contained, serverless, and zero-configuration database engine commonly embedded into mobile and desktop applications for local storage and capable of handling smaller-scale database needs with ease.

Deployment and Setup
: Extremely lightweight and easy to set up, with no server to install or configure.

Concurrent Access
: Generally supports single-user systems better, and might encounter locks or bottlenecks with multiple concurrent accesses.

Storage and Performance
: Optimized for local storage with smaller databases, with a file-based storage system that might be less efficient for large datasets or complex queries.

Translating logical designs to SQL

Relational Database Model

- **Tables** represent the fundamental entity for manipulation.
- Tables have multiple columns called **fields**, each with unique names and data types. **Fields** correspond to attributes in our ER model.
- Tables have multiple rows called **records** that represent individual instances of the data stored in the table. These are sometimes called *tuples*.
- Relationships between tables are documented as **keys**. Keys are simply fields in a table, designated as keys. Keys have special properties and the database keeps track of them differently.
- A complete definition entities, attributes and relations, along with any data types and constraints is called a **schema**.

```sql
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    name VARCHAR(100),
    year INT
);

CREATE TABLE courses (
    course_id INT PRIMARY KEY,
    title VARCHAR(100),
    credit_hours INT
);

CREATE TABLE student_courses (
    student_id INT,
    course_id INT,
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id),
    PRIMARY KEY (student_id, course_id)
);
```

In the example above, we define two entities: students and courses, and establish a many-to-many relationship between them. The relationships are documented in a separate table with foreign keys. The individual rows in the student-courses table represent pairs of students and courses or "student-course tuples".

In the coming weeks

- We'll spend the next lecture exploring how to translate specific logical designs into actual SQL.
- We'll also explore the fundamentals of relational algebra and how we tersly communicate operatings that we can perform on the data and how those operations look in SQL.
- Finally, we'll examine ways to minimize duplication of data by *functionally* analyzing our tables and evaluating it's *normal form*.

Housekeeping

- Homework 3 due Tuesday (tomorrow), 9/12
- Project deliverable 3 (video) due Friday.
- Check your grades to verify that we've got it all correct!
- Verify your group assignments, needed for deliverable 3!

Module	Week	Date	Day	Lectures/Quizzes	Deliverables/Notes
Relational Alg.	4	9/11	Mon	MTG5: L4 (DDL / DML / SQLLite / MySQL)	
Relational Alg.	4	9/12	Tue		HW3 due (ER Models 2 - extended)
Relational Alg.	4	9/13	Wed	MTG6: L5 (Relational models)	
Relational Alg.	4	9/15	Fri		PrjDel 3 due (Topic proposal video)
Relational Alg.	5	9/18	Mon	MTG7: L6 (Relational Algebra 1)	
Relational Alg.	5	9/20	Wed	MTG8: L7 (Relational Algebra 2)	
Relational Alg.	5	9/24	Sun		HW4 due (Relational Algebra Study Guide)
Normal forms	6	9/25	Mon	MTG9: Quiz 2 today (Relational Algebra)	