B561
Advanced Database Concepts
Fall 2022 Syllabus

Dirk Van Gucht[1]

[1] Indiana University

Syllabus overview

- Course main topics
- Course disclaimers (what this course is not about)
- Instructor and Associate Instructors
- Communication
- Exams (midterm and final)
- Assignments, policies, and grading protocol
- Exams and assignments grading schema (25% for midterm, 25% for final, 50% for assignments)
- Prerequisite knowledge
- Textbook and PostgreSQL system
- Lecture notes and associated program code
- List of lectures
- Indiana University policies

Course Main Topics

- Database models and systems: relational; nested and object-relational; semi-structured (JSON document databases); key-value stores; graph databases.
- State-changing database operations: database updates and triggers
- Query and database database programming languages: (object-relational) SQL, key-valued, partitioned, and distributed programming (e.g. Spark), property-graph querying
- Aspects and components of query processing: query transformation, translation, analysis, and optimization)
- Data structures and algorithms for efficient query processing
  - External sorting
  - Indexes: B^+-trees and hashing
  - Algorithms for a spectrum of fundamental query operations
- Elements of transaction management: concurrency and recovery

Disclaimer about course scope

- B561 is about database concepts which are designed from first principles and is elf-contained. It is more theoretical than systems-oriented.

  In particular, B561 is not a course that delves into the implementation of database management systems.

- Some of you may already be familiar with some or most of the course's concepts and therefore this course may not be for you. However, in most cases, the concepts will be handled at a more advanced and in-depth level.

- Importantly, this course is not designed to introduce you to a plethora of specific database system or database programming environments. Rather, it is designed such that the covered concepts are general and can be applied and deployed in other database environments. The actual deployment in such systems, other than in the PostgreSQL system, will not be done.

Disclaimer about course's scope (continued)

- B561 is also not a course on big data or distributed/parallel data processing nor on the development of domain-specific databases, projects, or database applications. If your aim is to learn about such aspects of databases and their use, other courses for such purposes and experiences exist in the School's course offerings.
- To understand what this course is about, I strongly recommend that you carefully look at the topics covered in lectures described below.
- B561 presupposes certain knowledge. Further down in this syllabus, I have listed what I assume this prerequisite knowledge to be.

Basic information

Instructor	Dirk Van Gucht
Email	vgucht@indiana.edu
Online office hours	Wednesday (8:30am-12:00pm)

Associate Instructors

TBA

Communication (IU Canvas announcements, Piazza, and email)

- Check the IU Canvas announcements daily (you should receive these in your email as well)
- Written communications about course and assignments **contents** will be through IUCanvas Piazza. You will need the access code B561Fall to use Piazza. Observe that such communications might be seen by all students. If a post and its answer are useful for the entire class it can be shared by me with the entire class. If I deem the post irrelevant or inappriopriate, I will delete it. I expect your posts to be bundled, i.e., if you have multiple questions for a topic or assignment, make every effort to bundle them. If not, I will not guarantee to answer each of your posts.
  - You can not post solutions or partial solutions to any assignment problems. You can also not post any issues related to due dates, grading, or personal issues.
  - For issues other than course contents and assignments, you can communicate with me via email.
- You can not assume that I will communicate with you over the weekends or during week-day evenings.

Exams

- There will be 2 exams: a midterm and a final
- Each exam will cover approximately 1/2 of the course material.

Exam schedule	
Midterm	October 18, 2022, 6:30–9:30pm
Final	During finals week
	Provisionally, December 12, 2022, 6:30-9:30pm

Assignments

- There will be 9 assignments. Most of these will have a theoretical and a programming component. Almost all of the programming will be done in the PostgreSQL system.
- Assignment 0 is designed to test your prerequisite knowledge. Solutions for it need not be submitted. It will not be graded.
- Assignment 1 through 7 will be graded with grades posted within approximately 10–14 days after they are due.
- Assignment 8 will not be graded. It is designed to cover the last lectures of the course. It is important to attempt to prepare for the final.
- Solutions for assignments will be posted shortly after their due dates.
- The solutions for the programming component of an assignment need to be submitted as a single file with a .sql extension and uploaded to Canvas before the due date. This .sql needs to compile correctly.
- The solutions for the theoretical component (when applicable) need to be submitted as a separate single file in .pdf format and uploaded to Canvas before the due date.

  I strongly encourage you to learn and use Latex since it is superior for mathematical text and such text will be needed in various assignments.

Assignments Policies

Late assignments will not be accepted. This policy is strict. I will ignore requests for extensions other than for extraordinary reasons. The AI's need to have the ability to start grading on time. It is therefore wise to submit your solutions or part of your solutions well before the due date. We will only grade assignments that are submitted prior to the deadline.

Collaboration: You can discuss problems on the assignments but all the people with whom you talk or communicate need to be identified clearly on your assignment submissions.[1] Failing to do so violates academic integrity policies. Solutions need to be in your own writing.

Plagiaralization software may be employed as provided by IU canvas.

[1] Note that each student is required to list these names which implies that cross-referencing of names can be checked.

Assignments grading protocol

The AI's will not grade all problems on the assignments. Rather, I will select a subset of the problems that they will be graded. For the non-graded problems, a default score will be given for each attempted problem. An attempted problem should be non-trivial.

The score for each assignment will be determined as follow:

- 70% of the score will be based on the graded problems
- 30% of the score will be based on the attempted problems (so attempting each problem guarantees a score of 30%)

In case you have questions about the grades on the assignments, first approach the AI who was the grader. If afterwards you still have questions, you can consult me.

In all cases though, before approaching us, you must have first checked the posted solutions and be able to relate your questions to these.

Exams and assignments grading schema

- Assignments carry a 50% weight towards the final grade. Different assignments may carry different weights. The weight of an assignment will be identified by the maximum points that can be obtained for it.
- The midterm exam and the final exam each carry a 25% weight towards the final grade.

I reserve the possibility to determine the final letter grades using a curve determined by the distribution of the numerical grades of all students.[2]

After the midterm exam, you will get an estimated letter grade for the class.[3]

Roughly speaking, I foresee that a grade around a ‘B’ will be awarded if your total score is around the average of all class scores.

---

[2] This is premised on whether the scores follow a normal distribution which is usually the case in this course.

[3] Caveat: IUCanvas automatically retains an estimated grade. This grade **should be ignored** because it is based on the recommended IU grading scale which is not the one I will adopt.

Prerequisite Knowledge

- **Programming**: It is assumed that you can program; ideally in various programming styles: imperative, functional, and object-oriented.
- **Mathematics**: Basic algebra and elements of Discrete Mathematics
- **Data Structures and Algorithms**

Knowledge required from Discrete Mathematics

Propositional Logic:

- Propositions
- Logical operators (and, or, not, implies)
- Truth assignments and truth tables
- Logical equivalences (e.g., laws of double negation, distribution, De Morgan, etc.)
- Inference rules and proofs

Knowledge required from Discrete Mathematics

Predicate logic:

- Domain variables (e.g, x, y, etc) and terms (e.g, x, $f(x, y)$)
- Predicates (e.g, $P(x)$, $Q(x, y)$, $R(x, y, z)$, etc.)
- Statements (e.g, $\forall y\, Q(x, y) \rightarrow P(x)$, etc.)
- Existential and universal quantifiers ($\exists\, x$, $\forall\, x$); free and bound variables;
- Set and relation abstraction (e.g, $\{x | \exists y\, P(x) \wedge Q(x, y)\}$)
- Predicate interpretations and truth values of statements
- Logical implication
- Logical equivalences, (e.g, $\neg \exists x\, P(x) \equiv \forall x\, \neg P(x)$, etc.)
- Inference rules and proofs

Knowledge required from Discrete Mathematics

- **Set theory**:
  - Operations on sets: union $\cup$, intersection $\cap$, difference $-$, cartesian product $\times$
  - Relations and functions; equivalence and order relations; one-to-one, onto, and correspondence functions
  - Set cardinality
- **Induction and recursion**:
  - Inductively defined sets and structures
  - Proofs by mathematical and structural induction(e.g, base cases; inductive cases)

Knowledge required from Data Structures and Algorithms

A good textbook on algorithms and data structures is *Data Structures and Algorithms* by Aho, Hopcroft, and Ullman

- **Data structures encountered in computing problems**: lists, arrays, stacks, queues, dictionaries, trees, (balanced) search trees, hash tables, graphs
- **Operations on data structures**: insert, delete, search, and reorganize
- **Asymptotic complexity analysis**: running time and space complexity of algorithms expressed in terms of $O(f(n))$

Knowledge required from Data Structures and Algorithms

- **Search**: Constant $O(1)$, linear $O(n)$, and logarithmic searching $O(\log(n))$
- **Merge**: merging of ordered lists L_1 and L_2 with complexity $O(|L_1| + |L_2|)$
- **Sorting** $O(n^2)$ sorting algorithms, merge sort $O(n \log(n))$
- **Tree and graph traversal**: breadth first seach (BFS) and depth first search (DFS); complexity is linear in the size of the tree or graph
- **Memory management**:
  - Allocation and deallocation of memory
  - Memory hierarchy: registers $\rightarrow$ cache $\rightarrow$ main memory (RAM) $\rightarrow$ secondary memory (disk)
  - Memory access time
  - Volatile and persistent memory

Textbook and PostgreSQL system

- **Textbook**: *Database Systems: The Complete Book (2nd Edition)* by Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom.

  In some cases, I will use other texts and sources.

  There will be lecture notes that are original and do not have a corresponding coverage elsewhere.

- **Database system**: we will use the PostgreSQL (version 13 or higher) system; you are required to install and test this system immediately on your personal computer; the system can be downloaded from the PostgreSQL site `https://www.postgresql.org/download/`

- **PostgreSQL manual**: you will need and be expected to frequently consult the PostgreSQL manual. This manual can be found at site `https://www.postgresql.org/docs/13/index.html`[4]

---

[4] Preface; Sections I and II; Section V 35.3, 35.4, 35.5, 37, 38, 41, 42; Section VII 51, 63, 68.

PostgreSQL

In this course, we will use PostgreSQL as our primary database management system.

PostgreSQL will be the database system to use in the programming assignments.

I decided to use PostgreSQL since it most cleanly and comprehensively captures the functionalities required to convey the concepts in this course.

You will notice in the lectures that we will frequently use these PostgreSQL functionalities directly to explain important concepts of the course.

PostgreSQL Installation

You will need to install PostgreSQL (version 13 or higher) on your personal computer.

This must be accomplished during the first days of the course. All assignments will require PostgreSQL.

For a Mac installation, visit site
`https://postgresapp.com/`

For a Windows installation, visit the site
`https://www.postgresql.org/download/windows/`

Lecture notes

There will be approximately 28 lectures covering the main topics for this course (for more details see below in the syllabus). Depending on the progress in the course, adjustments to the schedule and lectures may occur. Typically, lectures will be released on a weekly basis. Each lecture will have the following:

- **Lecture notes**: for each lecture, there will be a document that accompanies the corresponding lecture.
- **Code fragments**: whenever a lecture includes SQL code, that code and its outputs will appear in a corresponding `.sql` file; this code has been tested and can be run in the PostgreSQL interpreter
- **Textbook material**: Where relevant, textbook material associated with that lecture will be specified; on occasion, papers will be supplied; references to the PostgreSQL manual may be given

Lecture topics

Below at the titles and topics for each lecture. It is possible that some adjustments will be made to these lectures.

Each item in the list below corresponds to a lecture. You can anticipate that the lectures will be given in the order in which they are listed. Note however that there is not a one-to-one mapping between lectures and class periods: some lectures may be bundled in one classs period and other may span class periods.

1. **The relational database model**: databases, relations, insert, delete, primary keys, foreign keys
2. **SQL part 1**: syntax, semantics, and time complexity of SQL queries over single and multiple relations; queries with subqueries; queries with set operations union, intersection, and difference
3. **SQL Part 2**: parameterized subqueries; queries with set predicates [NOT] IN, ALL, SOME, and [NOT] EXISTS

Lecture topics

4. **The Tuple Relational Calculus (TRC) and SQL**: TRC is a formal logic to express relational queries and constraints; (Pure) SQL is equivalent with TRC and can thus also be viewed as a logic; TRC equivalences can be exploited to formulate SQL queries and constraints in different ways; reasoning about SQL can be reduced to reasoning about TRC.
5. **Views**: view definition; utilization of views in queries; view rewriting; updating views; materialized views; temporary views; parameterized views; recursive views

6. **Relational Algebra (RA)**: syntax of relational algebra (RA) as expressions with operations *selection* σ, *projection* π, *cross product* $\times$, *union* $\cup$, *intersection* $\cap$, and *difference* $-$; semantics of RA in SQL; expressing queries in RA

Lecture topics

7. **(Regular) Joins and semi-joins**: joins $\bowtie$ as operations to discover relationships between data objects; semi-joins $\ltimes$ as operations to discover properties of data objects; expressing joins and semi-joins in RA; expressing joins and semi-joins using SQL INNER JOIN, NATURAL JOIN, and CROSS JOIN operations
8. **Queries with quantifiers Part 1**: queries with some, no, [not] only, [not] all, at least k quantifiers; method of Venn-diagrams with condition to express queries with quantifiers; expressing queries with quantifiers in SQL with set predicates or set operations
9. **Set joins and set-semijoins** Set joins and set-semijoins generalize the types of regular joins and semi-joins that we consider in the previous lecture. Such joins enable expressing queries with quantifiers in the relational algebra.

Lecture topics

11. **Translation of SQL queries into RA**: techniques and algorithm to translate SQL queries into equivalent RA expressions
12. **Query optimization**: rewrite rules to transform RA expressions and SQL queries into optimized RA expressions and SQL queries; query-evaluation efficiency gains due to query optimization

Lecture topics

13. **SQL functions and expressions**: simple and complex expressions on tuple components in WHERE and SELECT clauses; boolean queries; user-defined SQL functions; functions with output parameters; functions returning tuples; functions returning relations; non-deterministic effects of functions;
    Introduction to **Dynamic SQL**: SQL query construction via string formulation; execution of SQL queries represented by strings.

Lecture topics

14. **Aggregate functions and data partitioning**: collection types; aggregate functions over collection types; applications of aggregate functions (data analytics, complex query formulation, efficient query evaluation); the COUNT aggregate function; other aggregate functions SUM, MIN, MAX, etc
    Data partitioning: grouping; parameterized queries; mapping aggregate functions over data partitions; GROUP BY method; count function method; count expression in SELECT clause method; grouping sets; data cubes; window functions

Lecture topics

15. **Queries with quantifiers Part 2**: This is a generalization of Part 1 wherein we consider queries with some, no, [not] only, [not] all, at least k quantifiers that return **pairs** of objects instead of just objects;
   Queries with Quantifiers using counting: translating queries with quantifiers as expressed by conditioned Venn diagrams into SQL queries with the COUNT aggregate function

Lecture topics

16. **Triggers**: functions that get invoked (triggered) when state-changes are applied to database relations and views; applications: view updates and materialization, constraint verification, event logging, and others

Lecture topics

17. **Physical database organization for efficient query processing**: disk and hardware organization for representing and querying databases;
External sorting: sorting in a two-tiered memory architecture; time complexity of external sorting

Lecture topics

18. Indexing
    - **Tree Indexing**: tree (B^+-tree) indexes and corresponding search, insert, and delete algorithms; time-complexity analysis
    - **Hash Indexing**: hash-based indexes and and corresponding search, insert, and delete algorithms; static and extensible hashing; time-complexity analysis

19. **Algorithms for RA operations**: algorithms for selections and projections; join algorithms: block-nested join, sort-merge join, and hash join; complexity analysis
20. **Query processing and query plans**: query plans that associated with RA SQL and Pure SQL queries; query planner parameters that affect the query plans; analysis of query plans for SQL queries with RA operators and SQL set predicates.

Lecture topics

21. **Object-relational databases and queries**: complex object types (composite types and array types); modeling sets with arrays; functions and predicates on complex objects; complex-object database restructuring; queries on complex-object databases.
22. **Nested relational databases, semi-structured databases, document databases**: nested relational database model; semi-structured database model using JSON objects; document-oriented databases; queries on such databases; programming around these will be accomplished in PostgreSQL.

Lecture topics

23. **Key-value stores. NoSQL in MapReduce Style**
    Discussion of the MapReduce distributed database programming model; key-value stores; mappers and reducers; simulations of MapReduce in object-relational SQL style
24. **Key-value stores. NoSQL in the SPARK distributed database modeling and query processing environment**
    Introduction and discussion of the Spark database programming model; key-value stores and RDDs (Resilient Distributed Datasets); Spark transformations and actions; simulation of Spark transformations and actions in object-relational SQL style

Lecture topics

25. **Graph databases and graph queries**: graph databases are semi-structured databases with node- and edge-labels; queries can be pattern-based or navigational, or a combination of both; such queries can be simulated in object-relational databases or can be specified in graph database management systems such as Neo4J

Lecture topics

26. **Object-relational database programming**: embedding of object-relational SQL code in a complete programming environment; recursive queries; queries requiring conditional and looping statements; procedures and functions

27. **Transaction management (concurrency)**: introduction to concurrency control; serial and serializable schedules; conflict serializability
28. **Transaction management (recovery)**: introduction to database recovery; logging techniques and recovery algorithms.

Indiana University Policies (Academic misconduct)

Academic conduct will be governed by the Indiana University policies found at the following sites:

https://studentcentral.indiana.edu/policies/index.html

Cheating of any kind on and assignment will result in reducing the overall grade by 1 letter grade.

Cheating of any kind on the midterm or final exam will result in an automatic F.

Each cheating incident will be reported to the University.