B561 Advanced Database Concepts: Syllabus

Dirk Van Gucht[1]

[1]Indiana University

January 12, 2020

Syllabus overview

- Course main topics
- Basic information
- Associate Instructors
- Exam schedule
- Assignments and assignment grading
- Grading schema
- Prerequisite knowledge
- Textbook and PostgreSQL system
- Lecture videos and lecture notes
- Lecture topics

Course Main Topics

- Database models and systems: especially relational, object-oriented, and semi-structured
- Query languages and aspects of database programming
- Database design and modeling
- Aspects and components of query processing
- Data structures and algorithms for efficient query processing
- Introduction to transaction management: concurrency and recovery

Course Main Topics (disclaimer about course's scope)

We wish to emphasize that this course is about database concepts. As such, the course is more theoretical than systems oriented. It is designed from first principles and is self-contained. Some of you may already be familiar with some or most of these and therefore this course may not be for you.
Specifically, this course is not designed to introduce you to a plethora of specific database system or database programming environments therein. Rather, it is designed in such a way that the covered concepts are generic and therefore can be deployed within these systems specific environments. The actual deployment in such systems, other than in PostgreSQL, will not be done.
It 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.
If you have questions about this, you are welcome to consult with me at the beginning of the course.

To understand what this course is about, I strongly recommend that you carefully look at the lectures described at the end of this syllabus.

Basic information

Instructor	Dirk Van Gucht
Email	vgucht@indiana.edu
Office	Luddy Hall 3054
Lectures	Released Mondays and Wednesdays
Residential office hours	Tues. and Thurs. (8:00-10:00 EST)
Online office hours	Tues. and Thurs. (10:15-12:15 EST)

Basic information: Office hours and Communication

Residential office hours will be in my office.
Online office hours will be conducted via Zoom (see instructions in Zoom module).

Communications about course contents and assignments will be through Piazza. Observe that such communications might be seen by all students. You can not post solutions or partial solutions to any assignment problems.

For issues other than course contents and assignments, you can communicate with me via email.

You should not assume that I can communicate with you over the weekends or during the week-day evenings.

Associate Instructors

Associate Instructor	email	Office hours
Vikraman Choudhury	vikraman@indiana.edu	TBA
Priscilla Chrislin	chpris@iu.edu	TBA
Elham Jafari	ejafari@indiana.edu	TBA
Gadugu Prahasan	prgadugu@iu.edu	TBA

For the online B561, all office hours will be conducted via Zoom.

Exam Schedule

There will be 3 exams.

Each exam will cover 1/3 of the course material.

Exam schedule	
Exam 1	Monday, February 17, 2020
Exam 2	Wednesday, March 25, 2020
Exam 3	Monday, May 4, 2020 (2:45-4:45)

Assignments

There will be 8-9 assignments. Most of these will have a theoretical and a programming component. Most of the programming will be in the PostgreSQL 12 system.

Except for the last assignment, these assignments will be graded and the grades will be posted within approximately 2 weeks after they are due.

Solutions for assignments will be posted shortly after their due dates.

Programming assignments need to be submitted as a single file with a .sql extension and uploaded to Canvas.

The solutions for the theoretical part will need to be submitted as a separate file in .pdf format and uploaded to Canvas.

Assignments Policies

Late assignments will not be accepted. (No exceptions.) 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.

You can discuss problems on the assignment but all the people with whom you talk or communicate need to be identified clearly on your assignment submissions. 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.

Assignment grading

The AIs will not grade all problems on the assignments.

Rather, I will select a subset of the problems that they will graded. For the non-graded problems, a default score will be given for each attempted problem.

The score for each assignment will be normalized to 100 points of which 70 points are for the graded problems and 30 points are for the non-graded problems.

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.

Grading scheme

- Assignments: 40% uniformly distributed over all assignments.
- Exams: 60% uniformly distributed over the 3 exams.

I reserve the possibility to determine the final letter grades using a curve determined by the distribution of the numerical grades of all students.[1]

After each exam, you will get an estimated letter grade for the class.[2] 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.

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

[2] 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**: 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

You can consult the notes on Propositional Logic on IUCanvas

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

You can consult the notes on Predicate Logic on IUCanvas

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)

You can consult the notes on Sets, Functions and Relations, and Recursion and Inductive Proofs on IUCanvas

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 also other texts and sources as reference and to read.)
- **Database system**: we will use the PostgreSQL (version 12) 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 to frequently consult the PostgreSQL manual. This manual can be found at site
  `https://www.postgresql.org/docs/12/index.html`

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 to convey 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 10) on your personal computer.

This must be accomplished during the first week of the course.

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

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

Lecture videos and lecture notes

There will be approximately 28 lectures covering the main topics for this course (for more details see below in the syllabus). These lectures will be released on a weekly basis– typically each Monday and Wednesday.

- **Video recordings**: for each lecture, a video recording will be made available; each lecture is typically delivered in several video components. There will be a couple of lectures for which no video recording will be supplied. These notes will need to be studied as supplied.
- **Lecture notes**: for each lecture, there will be a document that contains the transcript of the corresponding video lecture
- **Code fragments**: whenever a lecture includes SQL code, that code and its outputs will appear in a corresponding `lecture-code.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 in a

Lecture topics

Below at the titles and topics for each lecture. It is possible that some small adjustments will be made to these lectures.
Each item below corresponds to a lecture. The lectures will be given in the order that they are listed.

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. **Views**: view definition; utilization of views in queries; view rewriting; updating views; materialized views; temporary views; parameterized views; recursive views
5. **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

Lecture topics

6. **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

7. **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
8. **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

9. **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

10. **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

11. **(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
12. **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

13. **Translation of SQL queries into RA**: techniques and algorithm to translate SQL queries into equivalent RA expressions
14. **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

15. **Database modeling**: modeling database schemas, constraints, and class hierarchies with entity-relationship (ER) diagrams; translating ER diagrams into relational database schemas
16. **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

Lecture topics

17. **Distributed database modeling and query processing Part 1**: discussion of the MapReduce distributed database programming model; key-value stores; mappers and reducers; simulations of MapReduce in object-relational SQL style
18. **Distributed database modeling and query processing Part 2**: 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

19. **Nested relational databases and semi-structured databases**: nested relational database model; semi-structured database model using JSON objects; graph databases; queries on such databases; programming around these will be accomplished in PostgreSQL
20. **Graph database 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.

Lecture topics

21. **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

Lecture topics

22. **Physical organization for efficient query processing**: disk and hardware organization for representing and querying databases; overview of indexing

Lecture topics

23. **External sorting**: sorting in a two-tiered memory architecture; time complexity of external sorting
24. **Tree Indexing**: tree (B^+-tree) indexes and corresponding search, insert, and delete algorithms; time-complexity analysis
25. **Hash Indexing**: hash-based indexes and and corresponding search, insert, and delete algorithms; static and extensible hashing; time-complexity analysis

Lecture topics

26. **Algorithms for RA operations**: algorithms for selections and projections; join algorithms: block-nested join, sort-merge join, and hash join; complexity analysis

Lecture topics

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

1. https:studentaffairs.indiana.edu/student-conduct/index.html
2. https://policies.iu.edu/policies/ua-03-sexual-misconduct/index.html