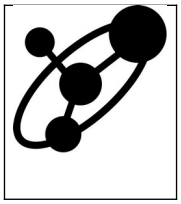
Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»



Software Engineering in
Energy Industry Department

GRAPH DATABASES

Syllabus

Catalog Description	
Higher education level	***Second (Graduate)***
Knowledge field	*Information Technologies*
Profession	*121 Software Engineering*
Curriculum	*Software Engineering of Intelligent Cyber-Physical Systems in Energy Industry*
Course status	*Elective*
Form of training	*Full-time*
Grade, term	*First grade, spring term*
Credits (hours)	*5 credits / 150 hours (full time: 36 hours of lectures, 28 hours of practice, 96 hr of individual assignments; part time: 10 hours of lectures, 6 hours of practice, 134 hours of individual assignments)*
Term control	*Exam, modular test*
Schedule	*http://schedule.kpi.ua/*
Teaching language	*Ukrainian/English*
Instructors	Lecturer: *DSc. (Econ), professor Andrii Sihaiov* Seminars: Laboratory work: *Andrii Sihaiov*
URL	

Course Program

1 Course description, aim, subject, and expected outcomes

Why future specialist should study this course?

Graph data closes the gap between the way humans and computers view the world. While computers rely on static rows and columns of data, people navigate and reason about life through relationships. This practical guide demonstrates how graph data brings these two approaches together. By working with concepts from graph theory, database schema, distributed systems, and data analysis, you'll arrive at a unique intersection known as graph thinking.

This course shows data engineers, data scientists, and data analysts how to solve complex problems with graph databases. You'll explore templates for building with graph technology, along with examples that demonstrate how teams think about graph data within an application.

Course Aim. *To familiarize students with graph data model and modern graph databases.*

Course Subject. *An overview of graph databases, including:*

- *Build an example of application architecture with relational and graph technologies*
- *Use graph technology to build a Customer 360 application, the most popular graph data pattern today*
- *Dive into hierarchical data and troubleshoot a new paradigm that comes from working with graph data*

- *Find paths in graph data and learn why your trust in different paths motivates and informs your preferences*
- *Use collaborative filtering to design a Netflix-inspired recommendation system*

Expected Outcomes.

Professional Competencies.

PC 13. Ability to implement applications using data and knowledge engineering concepts.

Program's Learning Outcomes.

PLO 17. Collect, analyze, evaluate information necessary for solving scientific and applied problems, using scientific and technical literature, databases and other sources.

PLO 21. Develop applications using data and knowledge engineering concepts.

2 Course prerequisites (Where the course fits into our curriculum)

The course is taken in the spring term of final year. Discrete Math and Databases are the prerequisites. There is no required course that has this course as a prerequisite.

3 Course contents

1. Graph Thinking

2. Evolving from Relational to Graph Thinking

3. Getting Started: A Simple Customer 360

4. Exploring Neighborhoods in Development

5. Exploring Neighborhoods in Production

6. Using Trees in Development

7. Using Trees in Production

8. Finding Paths in Development

9. Finding Paths in Production

10. Recommendations in Development

11. Simple Entity Resolution in Graphs

12. Recommendations in Production

13. Conclusion

4 Course textbooks and materials

Required reading:

Gosnell, D., Broecheler, M. The Practitioner's Guide to Graph Data: Applying Graph Thinking and Graph Technologies to Solve Complex Problems: Sebastopol, CA: O'Reilly Media, 2020. 420 c. URL: http://libgen.rs/book/index.php?md5=2F852C74D4139268D520CB3E4B1662D3

Optional reading:

1. *Bechberger, D., Perryman, J. Graph Databases in Action: Shelter Island, NY: Manning Publications, 2020. 336 c. URL: http://libgen.rs/book/index.php?md5=7F8919C423C42D10A3BBEAABF5026F3B*
2. *Needham, M., Hodler, A. E. Graph Algorithms: Practical Examples in Apache Spark and Neo4j: Sebastopol, CA: O'Reilly Media, 2019. 256 c. URL: http://libgen.rs/book/index.php?md5=cc42a7b9970c7f4930f014ababe5e03a*

3. *Lee, V., Nguyen, P. K., Thomas, A. Graph-Powered Analytics and Machine Learning with TigerGraph: Driving Business Outcomes with Connected Data: Sebastopol, CA: O'Reilly Media, 2023. 314 c. URL: http://libgen.rs/book/index.php?md5=375C33F4657BAD2EBC7F76A2E2BC842D*

Educational Content

5 Pedagogical advice

1. *Graph Thinking.*
   - *1.1. Why Now? Putting Database Technologies in Context.*
     - *1.1.1. 1960s–1980s: Hierarchical Data.*
     - *1.1.2. 1980s–2000s: Entity-Relationship.*
     - *1.1.3. 2000s–2020s: NoSQL.*
     - *1.1.4. 2020s–?: Graph.*
   - *1.2. What Is Graph Thinking?*
     - *1.2.1. Complex Problems and Complex Systems.*
     - *1.2.2. Complex Problems in Business.*
   - *1.3. Making Technology Decisions to Solve Complex Problems.*
     - *1.3.1. So You Have Graph Data. What's Next?*
     - *1.3.2. Seeing the Bigger Picture.*
   - *1.4. Getting Started on Your Journey with Graph Thinking.*
2. *Evolving from Relational to Graph Thinking.*
   - *2.1. Lecture Preview: Translating Relational Concepts to Graph Terminology.*
   - *2.2. Relational Versus Graph: What's the Difference?*
     - *2.2.1. Data for Our Running Example.*
   - *2.3. Relational Data Modeling.*
     - *2.3.1. Entities and Attributes.*
     - *2.3.2. Building Up to an ERD.*
   - *2.4. Concepts in Graph Data.*
     - *2.4.1. Fundamental Elements of a Graph.*
     - *2.4.2. Adjacency.*
     - *2.4.3. Neighborhoods.*
     - *2.4.4. Distance.*
     - *2.4.5. Degree.*
   - *2.5. The Graph Schema Language.*
     - *2.5.1. Vertex Labels and Edge Labels.*
     - *2.5.2. Properties.*
     - *2.5.3. Edge Direction.*
     - *2.5.4. Self-Referencing Edge Labels.*

2.5.5. Multiplicity of Your Graph.

2.5.6. Full Example Graph Model.

2.6. Relational Versus Graph: Decisions to Consider.

2.6.1. Data Modeling.

2.6.2. Understanding Graph Data.

2.6.3. Mixing Database Design with Application Purpose.

2.7. Summary.

3. Getting Started: A Simple Customer 360.

3.1. Lecture Preview: Relational Versus Graph.

3.2. The Foundational Use Case for Graph Data: C360.

3.2.1. Why Do Businesses Care About C360?

3.3. Implementing a C360 Application in a Relational System.

3.3.1. Data Models.

3.3.2. Relational Implementation.

3.3.3. Example C360 Queries.

3.4. Implementing a C360 Application in a Graph System.

3.4.1. Data Models.

3.4.2. Graph Implementation.

3.4.3. Example C360 Queries.

3.5. Relational Versus Graph: How to Choose?

3.5.1. Relational Versus Graph: Data Modeling.

3.5.2. Relational Versus Graph: Representing Relationships.

3.5.3. Relational Versus Graph: Query Languages.

3.5.4. Relational Versus Graph: Main Points.

3.6. Summary.

3.6.1. Why Not Relational?

3.6.2. Making a Technology Choice for Your C360 Application.

4. Exploring Neighborhoods in Development.

4.1. Lecture Preview: Building a More Realistic Customer 360.

4.2. Graph Data Modeling 101.

4.2.1. Should This Be a Vertex or an Edge?

4.2.2. Lost Yet? Let Us Walk You Through Direction.

4.2.3. A Graph Has No Name: Common Mistakes in Naming.

4.2.4. Our Full Development Graph Model.

4.2.5. Before We Start Building.

4.2.6. Our Thoughts on the Importance of Data, Queries, and the End User.

4.3. Implementation Details for Exploring Neighborhoods in Development.

4.3.1. Generating More Data for Our Expanded Example.
4.4. Basic Gremlin Navigation.
4.5. Advanced Gremlin: Shaping Your Query Results.
4.5.1. Shaping Query Results with the project(), fold(), and unfold() Steps.
4.5.2. Removing Data from the Results with the where(neq()) Pattern.
4.5.3. Planning for Robust Result Payloads with the coalesce() Step.
4.6. Moving from Development into Production.
5. Exploring Neighborhoods in Production.
5.1. Lecture Preview: Understanding Distributed Graph Data in Apache Cassandra.
5.2. Working with Graph Data in Apache Cassandra.
5.2.1. The Most Important Topic to Understand About Data Modeling: Primary Keys.
5.2.2. Partition Keys and Data Locality in a Distributed Environment.
5.2.3. Understanding Edges, Part 1: Edges in Adjacency Lists.
5.2.4. Understanding Edges, Part 2: Clustering Columns.
5.2.5. Understanding Edges, Part 3: Materialized Views for Traversals.
5.3. Graph Data Modeling 201.
5.3.1. Finding Indexes with an Intelligent Index Recommendation System.
5.4. Production Implementation Details.
5.4.1. Materialized Views and Adding Time onto Edges.
5.4.2. Our Final C360 Production Schema.
5.4.3. Updating Our Gremlin Queries to Use Time on Edges.
5.5. Moving On to More Complex, Distributed Graph Problems.
5.5.1. Our First 10 Tips to Get from Development to Production.
6. Using Trees in Development.
6.1. Lecture Preview: Navigating Trees, Hierarchical Data, and Cycles.
6.2. Seeing Hierarchies and Nested Data: Three Examples.
6.2.1. Hierarchical Data in a Bill of Materials.
6.2.2. Hierarchical Data in Version Control Systems.
6.2.3. Hierarchical Data in Self-Organizing Networks.
6.2.4. Why Graph Technology for Hierarchical Data?
6.3. Finding Your Way Through a Forest of Terminology.
6.3.1. Trees, Roots, and Leaves.
6.3.2. Depth in Walks, Paths, and Cycles.
6.4. Understanding Hierarchies with Our Sensor Data.
6.4.1. Understand the Data.
6.4.2. Conceptual Model Using the GSL Notation.
6.4.3. Implement Schema.

6.4.4. Before We Build Our Queries.

6.5. Querying from Leaves to Roots in Development.

6.5.1. Where Has This Sensor Sent Information To?

6.5.2. From This Sensor, What Was Its Path to Any Tower?

6.5.3. From Bottom Up to Top Down.

6.6. Querying from Roots to Leaves in Development.

6.6.1. Setup Query: Which Tower Has the Most Sensor Connections So That We Could Explore It for Our Example?

6.6.2. Which Sensors Have Connected Directly to Georgetown?

6.6.3. Find All Sensors That Connected to Georgetown.

6.6.4. Depth Limiting in Recursion.

6.7. Going Back in Time.

7. Using Trees in Production.

7.1. Lecture Preview: Understanding Branching Factor, Depth, and Time on Edges.

7.2. Understanding Time in the Sensor Data.

7.2.1. Final Thoughts on Time Series Data in Graphs.

7.3. Understanding Branching Factor in Our Example.

7.3.1. What Is Branching Factor?

7.3.2. How Do We Get Around Branching Factor?

7.4. Production Schema for Our Sensor Data.

7.5. Querying from Leaves to Roots in Production.

7.5.1. Where Has This Sensor Sent Information to, and at What Time?

7.5.2. From This Sensor, Find All Trees up to a Tower by Time.

7.5.3. From This Sensor, Find a Valid Tree.

7.5.4. Advanced Gremlin: Understanding the where().by() Pattern.

7.6. Querying from Roots to Leaves in Production.

7.6.1. Which Sensors Have Connected to Georgetown Directly, by Time?

7.6.2. What Valid Paths Can We Find from Georgetown Down to All Sensors?

7.7. Applying Your Queries to Tower Failure Scenarios.

7.7.1. Applying the Final Results of Our Complex Problem.

7.8. Seeing the Forest for the Trees.

8. Finding Paths in Development.

8.1. Chapter Preview: Quantifying Trust in Networks.

8.2. Thinking About Trust: Three Examples.

8.2.1. How Much Do You Trust That Open Invitation?

8.2.2. How Defensible Is an Investigator's Story?

8.2.3. How Do Companies Model Package Delivery?

8.3. Fundamental Concepts About Paths.

8.3.1. Shortest Paths.

8.3.2. Depth-First Search and Breadth-First Search.

8.3.3. Learning to See Application Features as Different Path Problems.

8.4. Finding Paths in a Trust Network.

8.4.1. Source Data.

8.4.2. A Brief Primer on Bitcoin Terminology.

8.4.3. Creating Our Development Schema.

8.4.4. Loading Data.

8.4.5. Exploring Communities of Trust.

8.5. Understanding Traversals with Our Bitcoin Trust Network.

8.5.1. Which Addresses Are in the First Neighborhood?

8.5.2. Which Addresses Are in the Second Neighborhood?

8.5.3. Which Addresses Are in the Second Neighborhood, but Not the First?

8.5.4. Evaluation Strategies with the Gremlin Query Language.

8.5.5. Pick a Random Address to Use for Our Example.

8.6. Shortest Path Queries.

8.6.1. Finding Paths of a Fixed Length.

8.6.2. Finding Paths of Any Length.

8.6.3. Augmenting Our Paths with the Trust Scores.

8.6.4. Do You Trust This Person?

9. Finding Paths in Production.

9.1. Chapter Preview: Understanding Weights, Distance, and Pruning.

9.2. Weighted Paths and Search Algorithms.

9.2.1. Shortest Weighted Path Problem Definition.

9.2.2. Shortest Weighted Path Search Optimizations.

9.3. Normalization of Edge Weights for Shortest Path Problems.

9.3.1. Normalizing the Edge Weights.

9.3.2. Updating Our Graph.

9.3.3. Exploring the Normalized Edge Weights.

9.3.4. Some Thoughts Before Moving On to Shortest Weighted Path Queries.

9.4. Shortest Weighted Path Queries.

9.4.1. Building a Shortest Weighted Path Query for Production.

9.5. Weighted Paths and Trust in Production.

10. Recommendations in Development.

10.1. Lecture Preview: Collaborative Filtering for Movie Recommendations.

10.2. Recommendation System Examples.

10.2.1. How We Give Recommendations in Healthcare.
10.2.2. How We Experience Recommendations in Social Media.
10.2.3. How We Use Deeply Connected Data for Recommendations in Ecommerce.
10.3. An Introduction to Collaborative Filtering.
10.3.1. Understanding the Problem and Domain.
10.3.2. Collaborative Filtering with Graph Data.
10.3.3. Recommendations via Item-Based Collaborative Filtering with Graph Data.
10.3.4. Three Different Models for Ranking Recommendations.
10.4. Movie Data: Schema, Loading, and Query Review.
10.4.1. Data Model for Movie Recommendations.
10.4.2. Schema Code for Movie Recommendations.
10.4.3. Loading the Movie Data.
10.4.4. Neighborhood Queries in the Movie Data.
10.4.5. Tree Queries in the Movie Data.
10.4.6. Path Queries in the Movie Data.
10.5. Item-Based Collaborative Filtering in Gremlin
10.5.1. Model 1: Counting Paths in the Recommendation Set.
10.5.2. Model 2: NPS-Inspired.
10.5.3. Model 3: Normalized NPS.
10.5.4. Choosing Your Own Adventure: Movies and Graph Problems Edition.
11. Simple Entity Resolution in Graphs.
11.1. Lecture Preview: Merging Multiple Datasets into One Graph.
11.2. Defining a Different Complex Problem: Entity Resolution.
11.2.1. Seeing the Complex Problem.
11.3. Analyzing the Two Movie Datasets.
11.3.1. MovieLens Dataset.
11.3.2. Kaggle Dataset.
11.3.3. Development Schema.
11.4. Matching and Merging the Movie Data.
11.4.1. Our Matching Process.
11.5. Resolving False Positives.
11.5.1. False Positives Found in the MovieLens Dataset.
11.5.2. Additional Errors Discovered in the Entity Resolution Process.
11.5.3. Final Analysis of the Merging Process.
11.5.4. The Role of Graph Structure in Merging Movie Data.
12. Recommendations in Production.

12.1. Lecture Preview: Understanding Shortcut Edges, Precomputation, and Advanced Pruning Techniques.

12.2. Shortcut Edges for Recommendations in Real Time.

12.2.1. Where Our Development Process Doesn't Scale.

12.2.2. How We Fix Scaling Issues: Shortcut Edges.

12.2.3. Seeing What We Designed to Deliver in Production.

12.2.4. Pruning: Different Ways to Precompute Shortcut Edges.

12.2.5. Considerations for Updating Your Recommendations.

12.3. Calculating Shortcut Edges for Our Movie Data.

12.3.1. Breaking Down the Complex Problem of Precalculating Shortcut Edges.

12.3.2. Addressing the Elephant in the Room: Batch Computation.

12.4. Production Schema and Data Loading for Movie Recommendations.

12.4.1. Production Schema for Movie Recommendations.

12.4.2. Production Data Loading for Movie Recommendations.

12.5. Recommendation Queries with Shortcut Edges.

12.5.1. Confirming Our Edges Loaded Correctly.

12.5.2. Production Recommendations for Our User.

12.5.3. Understanding Response Time in Production by Counting Edge Partitions.

12.5.4. Final Thoughts on Reasoning About Distributed Graph Query Performance.

13. Epilogue.

13.1. Where to Go from Here?

13.1.1. Graph Algorithms

13.1.2. Distributed Graphs

13.1.3. Graph Theory

13.1.4. Network Theory

13.2. Stay in Touch

6 Individual Assignments

Students expected to spend 3-4 hours a week outside of class on course material.

Course Rules and Assessment Policy

7 Course study rules

Students receive points relative to the correct and timely completion of coursework. The total grade consists of: 1) laboratories (programming assignments) 60%, 2) final exam 40%.
Presently there are three programming assignments, each worth up to 20% of the total grade. Student have to submit correctly fulfilled assignment during fortnight period from the date of give out to obtain full score for it, otherwise penalty points are applied but not more than 40% of the total score for laboratory work.

8 Assessment policy

How students are assessed: modular test, programming assignments
Calendar control: conducted twice a term to monitor the current state of compliance with the requirements of the syllabus.
Term assessment: final exam
Admission condition of term assessment: all programming assignment submission, start score not less than 40 points.
Exam scores map to the course grade according to the table:

Score	*Grade*
100-95	Excellent
94-85	Very Good
84-75	Good
74-65	Satisfactory
64-60	Sufficient
Less than 60	Unsatisfactory
Conditions for exam admission not met	Not allowed

9 Additional topics

- *Exam questions (see appendix).*

Syllabus:

Developed by Software Engineering in Energy Industry Department Professor, Sc. D. Andrii Sihaiov

Approved by Software Engineering in Energy Industry Department (minutes #28 on May 15, 2023)

Endorsed by Methodical Commission of Heat Power Faculty (minutes #9 on May 26, 2023)