Reading guidelines and key concepts

The following reading guidelines are divided into the three course parts/modules with a short introduction. Please consider this page and prepare yourself before the lectures and the lab/homework- exercises. See the guidelines below. You can both read in advance before the lectures and check out concepts that you did not understand during the lectures. The material is fully based on the course book.

This course uses the textbook:

H. Garcia-Molina, J. D. Ullman and J. Widom, Database Systems: The Complete Book (Second Edition), Pearson. (2009 or 2013 or 2014) ISBN (978-0131873254, 978-0-13-135428 or 978-1-292-02447-9)

The overall aim of the course is to equip you with the skills to model and implement database solutions. The students participating in the course are expected to take part in all activities on the course with a particular emphasis on the exercises and laboratories.

Make sure to understand the main concepts from each chapter and especially the concepts emphasized on the lectures.

Introduction

Chapter 1 - The Worlds of Database Systems

Reading:

1 - 1.1.3, 1.2.4, 1.3 (until Part III) (Read briefly)

Key concepts:

ACID: *Atomicity, Consistency, Isolation and Durability. See section 1.2.4 for further explanation.*

Database management system (DBMS): *A powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely.*

Part I - Relational Database Modeling

Chapter 2 - The Relational Model of Data

Reading:

2 - 2.4.11 (Read carefully)

2.4.12 - 2.4.13 (Optional reading)

2.5 - 2.5.4 (Read carefully)

Key concepts:

Data model: *A notation for describing data or information. A data model generally consist of three parts: Structure, operations and constraints on data.*

The relational model: *A table based model. Represent data as a two-dimensional table called a relation. The schema is declared separately from the data.*

The semi-structured-data model: *Resembles trees or graphs. The schema is implied by the data.*

Data-Definition sublanguage: *Used for declaring database schemas.*

Data-Manipulation sublanguage: *Used for querying databases and for modifying the database.*

SQL: *The principal query language for relational database systems.*

Relational Algebra: *Used for modeling the data stored in relational databases, and defining queries on it.*

Tables: *Stored relations.*

Temporary tables: *Constructed by the SQL language processor when it performs its job of executing queries and data modifications. (not stored). More in chapter 8*

Views: *Relations defined by a computation (not stored). More in chapter 8*

Relational database schema (database schema): *The set of schemas for the relations of a database.*

Schema (for a relation): *The name of a relation and the set of attributes for a relation.*

Attributes: *Describes the meaning of entries in the column below.*

Tuples: *The rows of a relation. (except the header row containing the attribute names)*

Components: *A container to hold a specific value. A tuple has one component for each attribute of the relation.*

Domains: *Associated with each attribute of a relation is a domain, that is, a particular elementary type (string, integer etc).*

Dangling tuple: *A tuple that fails to pair with any tuple of the other relation in a join.*

Constraints: *Used to specify rules for the data that may be stored in a database.*

Keys (primary key): *A set of attributes forms a key for a relation if we do not allow two tuples in a relation instance to have the same values in all the attributes of the key.*

Foreign Key: *An attribute or a collection of attributes (unique) of one relation can be a foreign key, referencing some attribute(s) of a second relation (more in ch 7).*

Referential-integrity constraints: *Requires that a value appearing in one column of one relation also appear in some other column of the same or a different relation.*

Chapter 3 - Design Theory for Relational Databases

Reading:

3 - 3.2.4 (Read carefully)

3.2.5 (Optional reading)

3.2.6 - 3.4 (Read carefully)

3.4.1 - 3.4.3 (Optional reading)

3.4.4 (Read carefully)

3.5 - 3.5.2 (Read carefully)

3.5.3 (Optional reading)

3.6.6 - Relationships Among Normal Forms (Read briefly)

Key concepts:

Normalization: *A process that removes anomalies by decomposition of a relation into two or more relations.*

Decomposition: *Breaking a relation into two smaller relations.*

Functional dependencies (FD): *A special relationship between two attributes. See Definition in section 3.1.1.*

Superkey: *A set of attributes within a relation whose values functionally determines all the other attributes in the relation.*

Minimal key (candidate key): *A minimal set of attributes within a relation whose values functionally determines all the other attributes in the relation.*

Trivial functional dependencies: *Has a right side that is a subset of its left side. See Definition in section 3.2.3.*

A closure of an attribute set A: *The set of all those attributes which can be functionally determined from the attribute set A.*

A basis for an FD's set A: *Any set of FD's equivalent to A.*

A minimal basis: *The minimal basis of a set of FD's with singleton right sides. See definition in section 3.2.7.*

Anomalies: *Problems such as redundancy that occur when we try to cram too much into a single relation. The principal kinds of anomalies: Redundancy, Update and Deletion anomalies. See explanation in section 3.3.1.*

Boyce-Codd normal form (BCNF): *A normal form that guarantees that the anomalies above does not exist and that the original relation could be recovered. See Definition in section 3.3.3.*

Third Normal Form (3NF): *A normal form that have both the lossless-join and dependency-preservation properties.*

Fourth Normal Form (4NF): *A normal form which condition is essentially the same as for BCNF, but applied to MVD's instead of FD's.*

Prime: *An attribute that is a member of some key.*

Chapter 4 - High-Level Database Models

Reading:

4 - 4.5.4 (Read carefully)

4.6 - 4.6.3 (Read briefly)

4.6.4 (Optional reading)

4.7 - 4.8.4 (Read carefully)

4.9 - 4.10.5 (Optional reading)

Key concepts:

Entity-relationship diagram (E/R-Diagram): *A notation for describing schemas of a database.*

Entity: *An abstract object of some sort. An entity in some ways resembles an object in the sense of object-oriented programming.*

Entity set: *A collection of similar entities forms an entity set.*

Attributes: *Properties of the entities in a specific set.*

Relationships: *Connections among two or more entity sets.*

Relationship set: *A set of relationships of similar type.*

Multiplicity of relationships: *many-many, many-one and one-one. See explanation in section 4.1.6.*

Subclasses: *There is a significant resemblance between isa in the E/R model and subclasses in object-oriented languages. In a sense, isa relates a subclass to its superclass. However,*

there is also a fundamental difference between the conventional E/R view and the object-oriented approach. See section 4.1.11.

Weak entity sets: *An entity set which key is composed of attributes where some or all of which belong to another entity set.*

Supporting entity sets: *Entity sets which have attributes that are inherited from a weak entity set.*

Supporting relationship: *The relationship that connects a supporting entity set with a weak entity set.*

Unified Modeling Language (UML): *A popular notation for describing database designs. Note: (It's also a graphical notation for describing software designs in an object-oriented style, like you did in the INDA course).*

Class: *A class in UML is similar to an entity set in the E/R model.*

Association: *A binary relationship between classes.*

Association Classes: *A class attached to the middle of an association with attributes associated with the relationship. See explanation in section 4.7.5.*

Complete: *Every object in one class is a member of some subclass.*

Partial/incomplete: *The opposite of complete.*

Disjoint: *An object cannot be in two of the subclasses .*

Overlapping: *An object can be in two or more of the subclasses.*

Aggregations: *A many-one association from the class at the opposite end to the class at the diamond end.*

Compositions: *Similar to aggregation but every object at the opposite end from the diamond must be connected to exactly one object at the diamond end.*

Part II - Relational Database Programming

Chapter 5 - Algebraic and Logical Query Languages

Reading:

5 - 5.1.6 (Read carefully)

5.2 - 5.2.3, 5.2.6 - 5.2.7 (Read briefly, learn the different kinds off operations. You don't have to study the relational algebra operators.)

5.2.4 - 5.2.5 (Optional reading)

Key concepts:

Bags (multiset): *Allows the same tuple to appear more than once in a relation.*

Chapter 6 - The Database Language SQL

Reading:

6 - 6.2.3 (Focus on learning the language)

6.2.4 (Optional reading)

6.3 - 6.5.3 (Focus on learning the language)

6.6 - 6.6.6 (Focus on learning the theory)

Key concepts:

NULL: *A special value and an important keyword. See section 6.1.6, 6.4.6.*

UNKNOWN: *The value in-between true and false in tree-way logic. See section 6.1.7.*

Subquery: *A query that is part of another query.*

Transactions: *A collection of one or more operations on the database that must be executed atomically.*

Read-only transactions: *A transaction that will never change the database.*

Isolation level: *See figure 6.17 in section 6.6.6.*

Serializability (default): *Transactions must behave as if they were run serially, one at a time.*

Repeatable read: *Tuples already selected are guaranteed to be unchanged during the transaction. Later queries can retrieve phantom tuples.*

Read committed: *Forbids the reading of dirty data.*

Read uncommitted: *The transaction is allowed to read dirty data.*

Phantom tuples: *Tuples that result from insertions into the database while our transaction is executing.*

Dirty data: *Data written by a transaction that has not yet committed.*

Dirty read: *A read of dirty data written by another transaction.*

Chapter 7 - Constraints and Triggers

Reading:

7 - 7.2.4 - (Read carefully)

7.3 - 7.5.1 - Triggers (Read briefly)

7.5.2 - The Options for Trigger Design (Optional reading)

Key concepts:

Active element: *An expression or statement that we write once and store in the database, expecting the element to execute at appropriate times.*

Modification (delete and update) policies: *See section 7.1.2.*

The Default Policy: *Any modification violating the referential integrity constraint is rejected.*

The Cascade Policy: *Changes to the referenced at tribute(s) are mimicked at the foreign key.*

The Set-Null Policy: *When a modification to the referenced relation affects a foreign-key value, the latter is changed to NULL.*

Not Deferrable (default): *Every time a database statement is executed, the constraint is checked immediately after each tuple modification.*

Deferrable Initially Deferred: *Wait until a transaction is complete before checking the constraint. Checking will be deferred to just before each transaction commits.*

Deferrable Initially Immediate*: The check will be made immediately after each statement in the transaction.*

Different kinds of constraints: *Not-Null Constraints, Attribute-based check constraint, Tuple-based check constraint.*

Comparisons of constraints: *Box in section 7.4.3.*

Assertion: *A boolean-valued SQL expression that must be true at all times.*

Trigger (ECA rules): *A series of actions that are associated with certain events, such as insertions into a particular relation, and that are performed whenever these events arise.*

Chapter 8 - Views and Indexes

Reading:

8 - 8.2.3 (Read briefly)

8.3 - 8.3.1 (Read carefully)

8.3.2 (Optional reading)

8.4 - 8.4.2 (Read carefully)

8.4.3 (Optional reading)

8.5 - 8.5.2 (Read carefully)

8.5.3 (Optional reading)

Key concepts:

Indexes: *A stored data structure whose sole purpose is to speed up the access to specified tuples of one of the stored relations.*

Materialized View: *Constructed periodically from the database and stored there.*

Virtual View: *Relations that are defined by a query over other relations (not stored).*

Chapter 9 - SQL in a Server Environment

Reading:

9 - 9.2.1 (Read briefly)

9.2.2 - 9.7.7 (Optional reading)

Part III - Modeling and Programming for Semistructured Data

Chapter 11 - The Semistructured-Data Model

Reading:

11 - 11.2.7 (Read carefully)

11.3, 11.4 (Read briefly)

11.4.1 - 11.4.7 (Optional reading)

Key concepts:

The legacy-database problem: *Once a database has been in existence for a while, it becomes impossible to disentangle it from the applications that grow up around it, so the database can never be decommissioned.*

Nodes: *The main elements of semistructured data. See section 11.1.2.*

Arcs: *Relates nodes to each other. See section 11.1.2.*

Elements: *A pair of matching tags and everything that comes between them.*

Attributes: *An alternative way to represent a leaf node of semistructured data.*

Namespaces: *Used to distinguish among different vocabularies for tags in the same document.*

Extensible Markup Language (XML): *Is a markup language that defines a set of rules for encoding documents (wiki).*

Well-formed XML: *Allows you to invent your own tags, much like the arc-labels in semistructured data.*

Valid XML: *Intermediate between the strict-schema models such as the relational model, and the completely schemaless world of semistructured data. Involves a DTD.*

Document Type Definition (DTD): *Specifies the allowable tags and gives a grammar for how they may be nested. - Section 11.3.*

XML Schema: *An alternative way to provide a schema for XML documents. It is more powerful than DTD's.*

Chapter 12 - Programming Languages for XML

Reading:

12 - 12.2.10 (Read carefully)

12.3 - 12.3.6 (Optional reading)

Key concepts:

XPath: *A simple language for describing sets of similar paths in a graph of semistructured data.*

XQuery: *An extension of XPath that adopts something of the style of SQL. It allows iterations over sets, subqueries, and many other features that will be familiar from the study of SQL.*

Items (XPath): *Values of a primitive type, Nodes - See section 12.1.1.*

Nodes: *Documents, XML Elements, Attributes.. - See section 12.1.1.*

Documents: *Files containing an XML document.*

FLWR Expressions: *The most important form of XQuery expression. Involves clauses of four types, called for-, let-, where-, and return- (FLWR) clauses.*