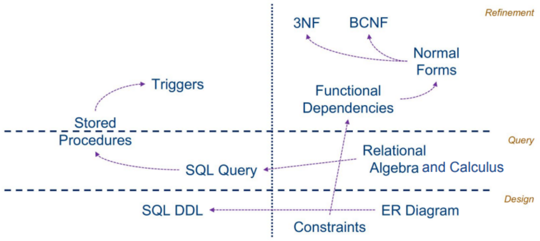
CS2102 Database Sys Summary

AY23/24 Sem 1, github.com/gerteck

Topics & Objectives

- **Design**: Entity-Relationship (ER) Model, Functional Dependencies, Normal Forms
- **Implementation**: SQL (Data definition language, Queries, Stored procedures, Triggers)
- **Theory**: Relational Calculus and algebra
- Module covers fundamental concpets and techniques for:
  - Understanding and practice of design & implementation of database applications and management of data with relational db management systems.
  - Design of ER data models to capture data requirements, translate to relational database schema, refine using schema decompositions to avoid anomalies.
  - Use SQL to define relational schemas, write queries.
  - Reason about correctness using concepts of formal query lang (relational calculus & algebra) and apply knowledge to develop database applications.



1. Database Management Sys DBMS

Challenges for Data-Intensive applications

- **Efficiency**: Fast access to information in volumes of data
- **Transactions**: ”All or nothing” changes to data
- **Data Integrity**: Parallel access and changes to data
- **Recovery**: Fast and reliable handling of failures (e.g. HDD/Sys crash, power outage, network disruption)
- **Security**: Fine-grained data access rights

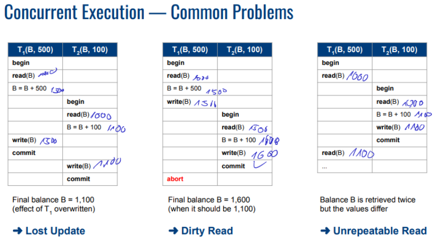
File-based data management to DBMS

- Complex, low level code, Often similar requirements across different programs
- **Problems**: High development effort, Long development times, Higher risk of (critical) errors
- **DBMS**: Set off universal and powerful functionalities for data management, with faster application development, higher stability, less errors.

Core concepts of DBMS

- **ACID Transaction**: Finite sequence of database operations (reads and/or writes), smallest logical unit of work
- **Atomicity**: either all effects of T are reflected in the database or none (”all or nothing”)
- **Consistency**: the execution of T guarantees to yield a correct state of the database
- **Isolation**: execution of T is isolated from the effects of concurrent transactions
- **Durability**: after commit of T, its effects are permanent even in case of failures

Concurrent Execution



Concurrent Execution — Common Problems

T_1(B, 500)	T_2(B, 100)
begin	
read(B) 1000	
B = B + 500 1500	
	begin
	read(B) 1000
	B = B + 100 1100
write(B) 1500	
commit	
	write(B) 1100
	commit

Final balance B = 1,100
(effect of T_1 overwritten)

➜ Lost Update

T_1(B, 500)	T_2(B, 100)
begin	
read(B) 1000	
B = B + 500 1500	
write(B) 1500	
	begin
	read(B) 1500
	B = B + 100 1600
	write(B) 1600
	commit ✓
abort	

Final balance B = 1,600
(when it should be 1,100)

➜ Dirty Read

T_1(B, 500)	T_2(B, 100)
begin	
read(B) 1000	
	begin
	read(B) 1000
	B = B + 100 1100
	write(B) 1100
	commit
read(B) 1100	
...	

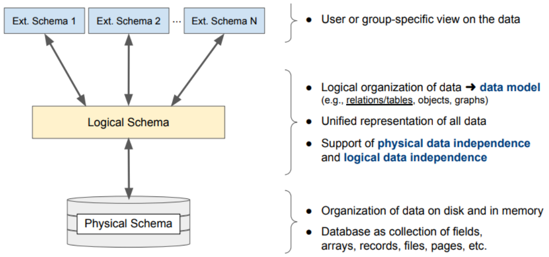
Balance B is retrieved twice
but the values differ

➜ Unrepeatable Read

Require Serializable transaction execution:

- A concurrent execution of a set of transactions is serializable if this execution is equivalent to some serial execution of the same set of transactions
- Two executions are equivalent if they have the same effect on the data
- **DBMS**: Support concurrent executions of transactions to optimize performance, Enforce serializability of concurrent executions to ensure integrity of data

Data Abstraction

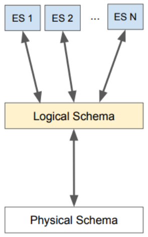


- User or group-specific view on the data

- Logical organization of data ➜ **data model** (e.g., relations/tables, objects, graphs)
- Unified representation of all data
- Support of **physical data independence** and **logical data independence**

- Organization of data on disk and in memory
- Database as collection of fields, arrays, records, files, pages, etc.

Data Independence



Logical data independence

- Ability to change logical schema without affecting external schemas (e.g., adding/deleting/updating attributes, changing data types, changing data model)

Physical data independence

- Representation of data independent from physical scheme
- Physical schema can be changed without affecting logical schema (e.g., creating indexes, new caching strategies, different storage devices)

Terminology / Definitions

- **Data Model**: Collection of concepts for describing data
- **Schema**: Description of structure of DB using data model
- **Schema Instance**: Content of a DB at a particular time

Relational Data Model

Data is modelled by relations, and each relation has a definition called a relation schema. This schema specifies attributes (columns) and data constrains (e.g. domain constraints)

- **Relation**: Can be seen as Tables with rows and columns:
  - No. of cols = Degree/Arity, No. of rows = Cardinality
  - Each row is called a tuple/record. It has a component for each attribute of the relation.
  - A relation is thus a set of tuples and an instance of the relation schema, i.e. of a single table.
- **Domain**: Set of atomic values, e.g. integers. All values for an attribute is either in this domain or null.
- **Relational database schema**: Set of relation schemas and their data constraints, i.e. of multiple tables
- **Relational database**: Instance of the schema and is a collection of tables.

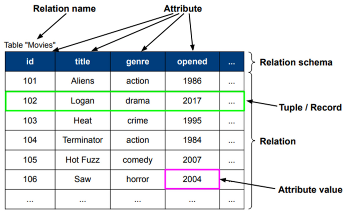
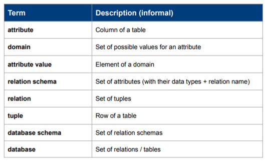


Table "Movies"

id	title	genre	opened	...
101	Aliens	action	1986	...
102	Logan	drama	2017	...
103	Heat	crime	1995	...
104	Terminator	action	1984	...
105	Hot Fuzz	comedy	2007	...
106	Saw	horror	2004	...
...	...	...	...	...

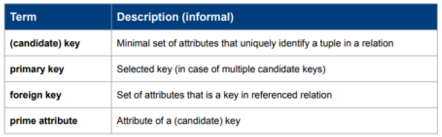


Term	Description (informal)
attribute	Column of a table
domain	Set of possible values for an attribute
attribute value	Element of a domain
relation schema	Set of attributes (with their data types + relation name)
relation	Set of tuples
tuple	Row of a table
database schema	Set of relation schemas
database	Set of relations / tables

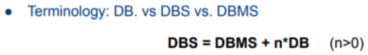
Integrity Constraints

Condition that restricts the data that can be stored in a database instance. A legal relation instance is a relation that satisfies all specified ICs.

- **Domain Constraints**: Restrict the attribute values of relations, e.g. only integers allowed
- **Key Constraints**:
  - **Superkey**: A superkey is a subset of attributes in a relation that unique identifies its tuples.
  - **Key**: A key is a superkey which is minimal, i.e. no proper subset of itself is a superkey.
  - **Candidate keys**: Set of all possible keys for a relation. One of these keys is selected as the primary key.
  - **Primary key**: Chosen candidate key for a relation, Cannot be null (entity integrity constraint), Underlined in relation schema. Prime attribute: Attribute of a primary key (cannot be null)
- **Foreign Key Constraints**:
  - **Foreign key**: A foreign key refers to the primary key of a second relation (which can be itself)
  - Each foreign key value must be the primary key value in the referenced relation or be null (foreign key constraint)
  - Also known as referential integrity constraints.



Term	Description (informal)
(candidate) key	Minimal set of attributes that uniquely identify a tuple in a relation
primary key	Selected key (in case of multiple candidate keys)
foreign key	Set of attributes that is a key in referenced relation
prime attribute	Attribute of a (candidate) key



- Terminology: DB. vs DBS vs. DBMS

DBS = DBMS + n*DB (n>0)