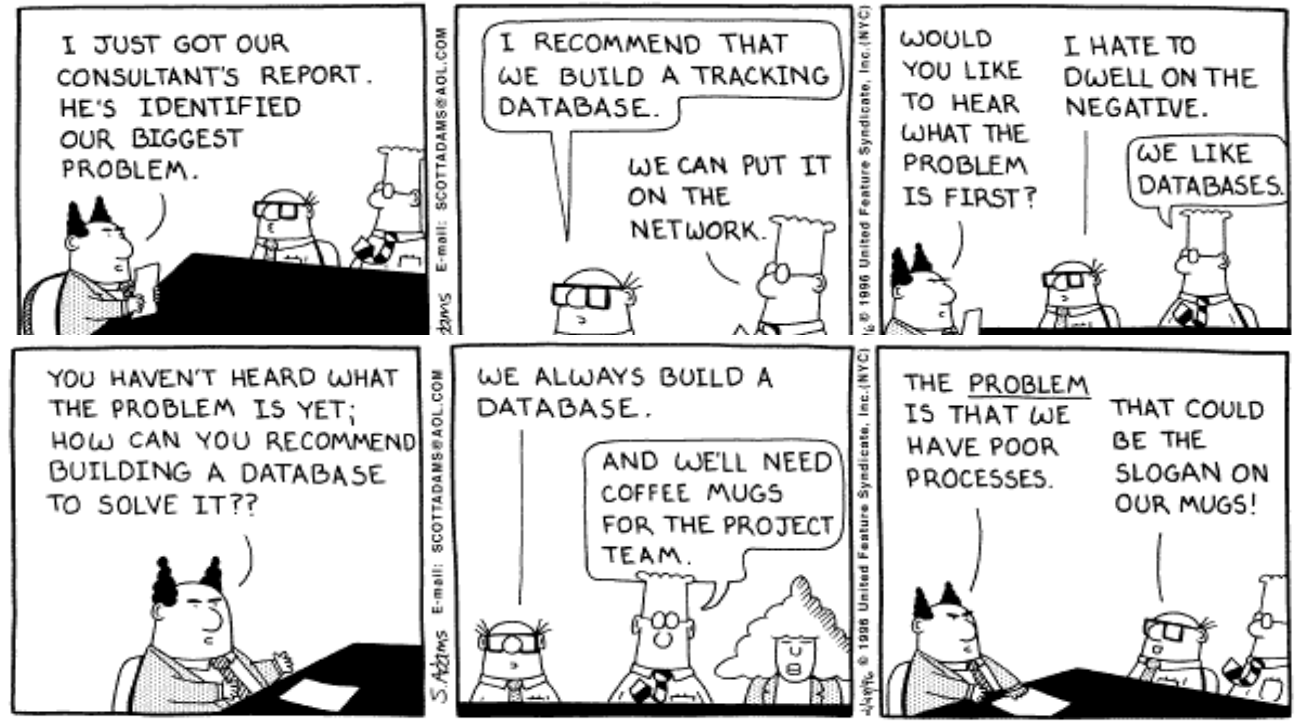
CS3223: Database Systems Implementation (https://www.comp.nus.edu.sg/~tankl/cs3223)



"Knowledge is of two kinds: we know a subject ourselves, or we know where we can find information upon it."

-- Samuel Johnson (1709-1784)

Course Admin

Course Instructor: Professor TAN Kian-Lee

Email: tankl@comp.nus.edu.sg

Office: COM1, #03-23

URL: https://www.comp.nus.edu.sg/~tankl/cs3223

TAs: TBD (see course webpage)

CS3223: A "Second" DB Course

CS2102 (Application)

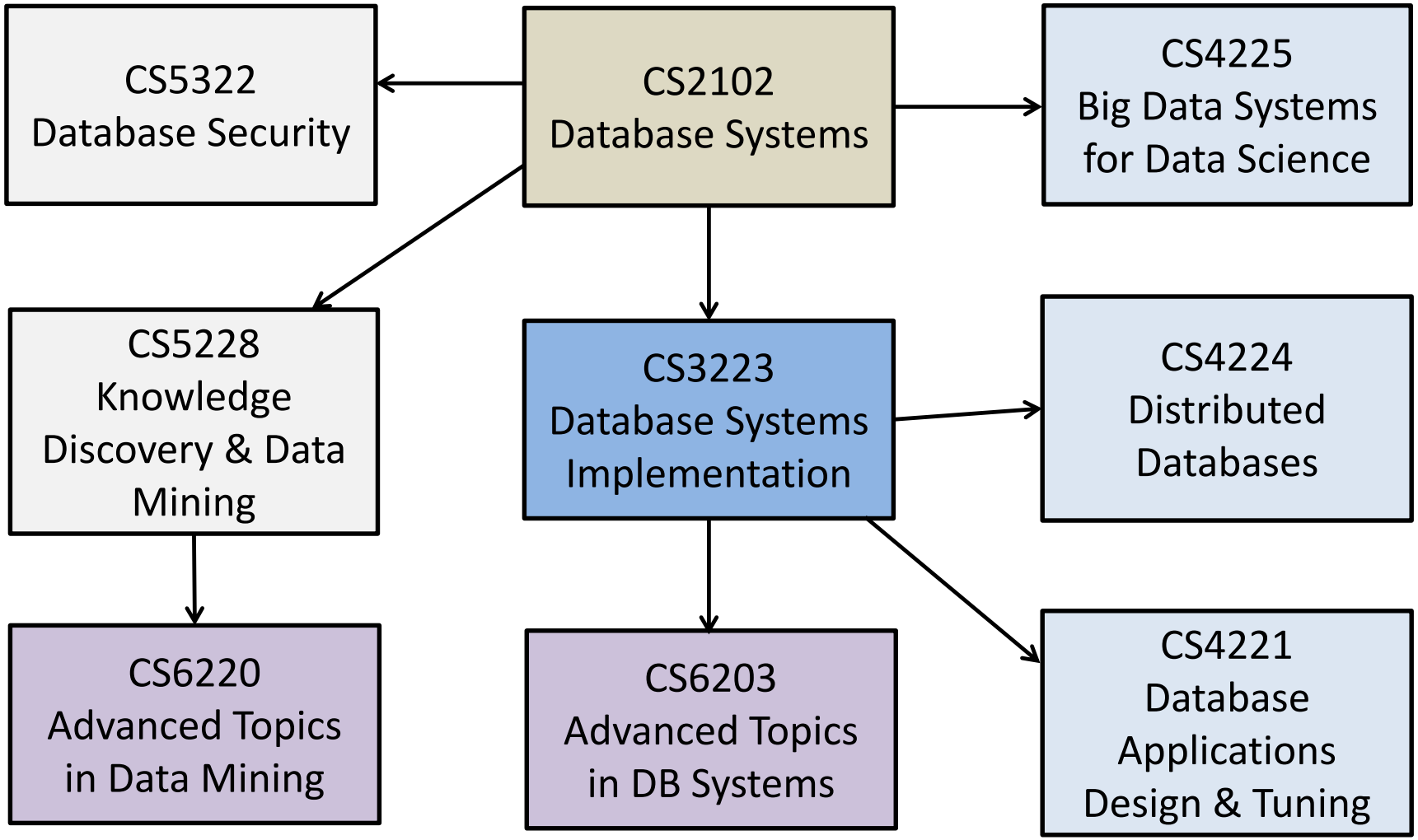
- Database design
  - ER design and normalization theory
  - Integrity management by design
  - Relational model
- Database programming
  - Relational algebra and calculus
  - Query languages
    - SQL
    - Embedded languages

CS3223 (Systems)

Database Implementation

- Storage management
  - Disk-based data organization for efficient data access
  - Indexes
  - Buffer management
- Query processing
  - Operators: Sort, join aggregates, etc
  - Query processing and optimization
- Transaction management
  - Concurrency control and recovery
- System Implementation Project

Database Courses @ SoC



Course Policies

- Students are responsible for the following:
  - Attend lectures & tutorials
  - Study referenced materials
  - Check course website/LumiNUS/emails for course-related announcements/updates
- Late assignment/project submissions will not be accepted without prior approval from lecturer
- Students can seek clarifications on lecture materials as follows:
  - Post questions to LumiNUS Discussion Forum
  - Email the lecturer
    - Emailed questions may be posted to LumiNUS Forum and answered there or emailed to all

Course Policies (cont.)

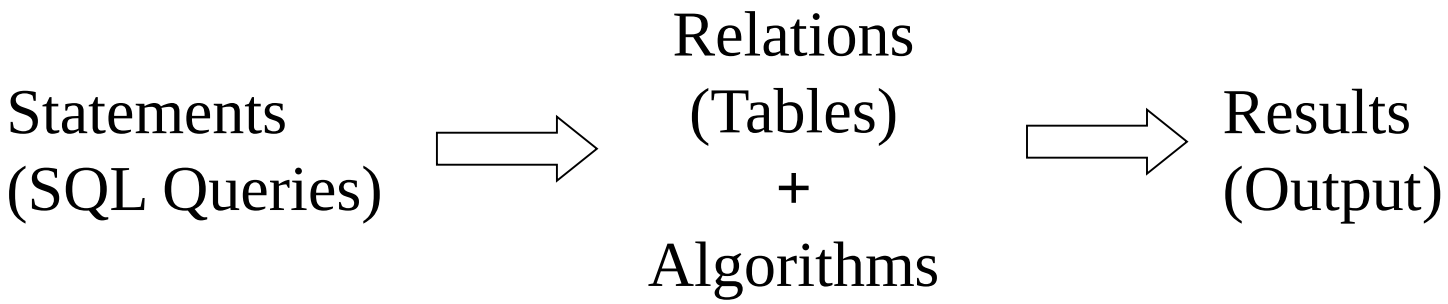
- **Zero-tolerance for plagiarism**

  https://www.comp.nus.edu.sg/cug/plagiarism/

- Plagiarism is generally defined as the practice of taking someone else's work or ideas and passing them off as one's own (The New Oxford Dictionary of English).
  - *You have the obligation to make clear to the assessor which is your own work, and which is the work of others*. Otherwise, your assessor is entitled to assume that everything being presented for assessment is being presented as entirely your own work. This is a minimum standard.
  - You may not knowingly intend to plagiarise, but that should not be used as an excuse for plagiarism. *You should seek clarification* from your instructors if you are unsure.

Introduction

Isn't Implementing a (Relational) Database (Management) System Simple?



myDBMS: A Simple Implementation

- Relations stored in files (ASCII), e.g., relation R is in /usr/db/R

```
Smith # 123 # CS
Jones # 522 # EE
```

- Directory file (ASCII) in /usr/db/directory

```
R1 # A # INT # B # STR …
R2 # C # STR # A # INT …
```

Sample evalutation strategy

- To execute "**`select * from R where R.a = 10`**":

  (1) Read dictionary to get R attributes

  * Exit if there is any error (e.g., R does not exists, R.a is not of type int)

  (2) Read R file, for each line:

  (a) Check condition: R.a = 10?

  (b) If OK, output/display record

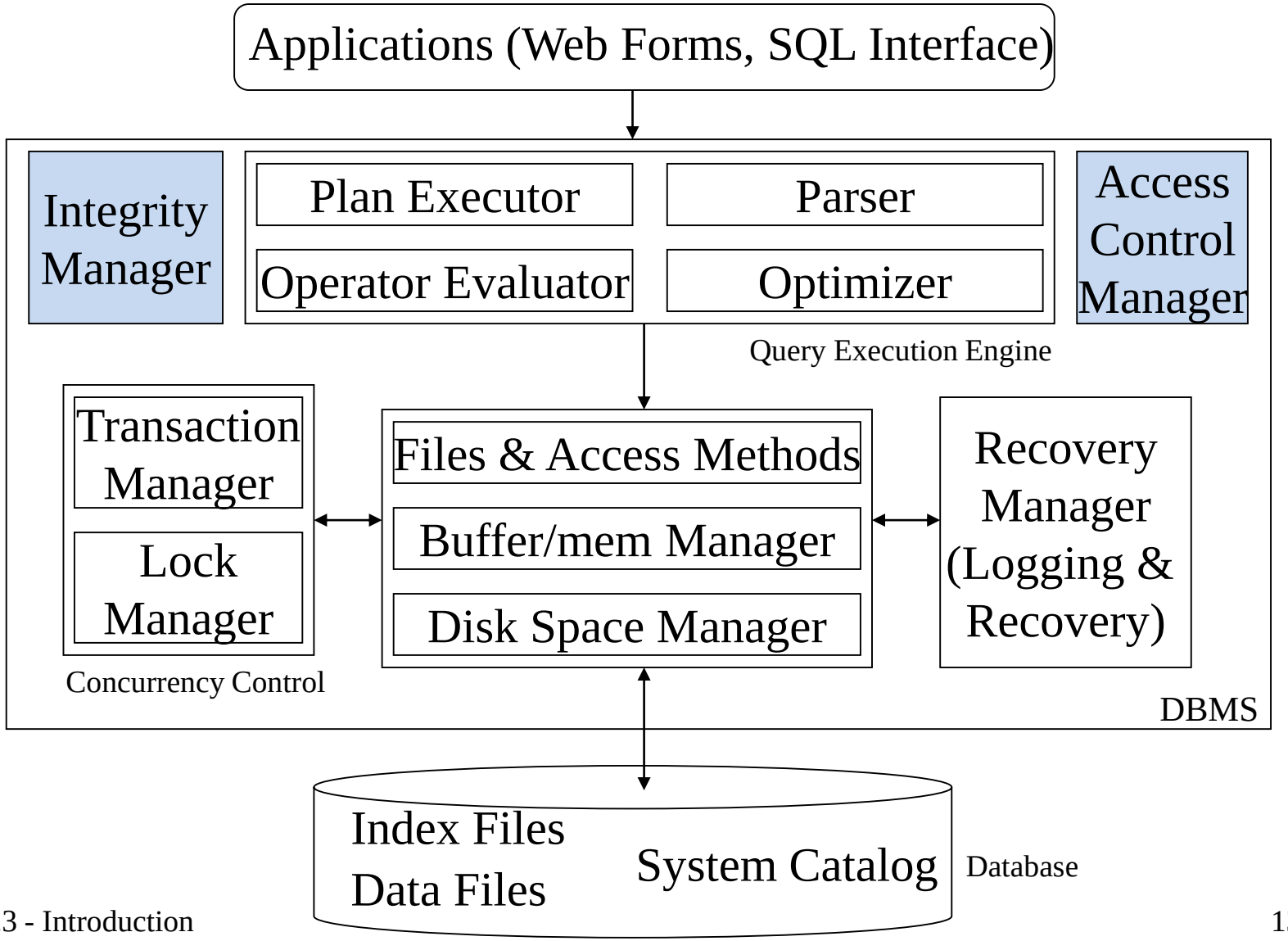
What's "wrong" with myDBMS?

- Tuple layout on disk
  - e.g., Change string from 'Cat' to 'Cats' and we have to rewrite file
  - ASCII storage is expensive
  - Deletions are expensive
- Search expensive; no indexes
  - Cannot find tuple with a given key quickly
  - Always have to read/scan full relation
- Brute force query processing

What's wrong with myDBMS?

- No buffer manager
  - Need caching
- No concurrency control
- No reliability
  - Can lose data
  - Can leave operations half done!
- No security
  - File system insecure
  - File system security is coarse
- No application program interface (API)
  - How can a payroll program get at the data?
- No GUI
- Poor dictionary facilities
- Cannot interact with other DBMS/tools

Architecture of a DBMS



Capabilities of a Modern (R)DBMS

- **Persistence** - permanent storage of data
- **Efficiency** - manage *large* volumes of data efficiently
- **High-level access** - data model & language for defining database structures, retrieval and manipulation
- **Transaction management** - provide correct, concurrent access to the database by many users at once
- **Access control** - limit access by unauthorized users
- **Integrity management** - assure compliance to known constraints imposed by application semantics
- **Resiliency** - ability to recover from system failures without losing data