Administrarea Bazelor de Date
Managementul în Tehnologia Informaţiei

# Sisteme Informatice şi Standarde Deschise (SISD)

## 2009-2010

## Curs 6

Sisteme de Gestiune a Bazelor de Date

# BASIC CONCEPTS

- **What is a database?**
  - A database is a collection of data which can be used: alone, or combined / related to other data to provide answers to the user's question.

- **What is a Database Management System?**
  - A DBMS is a collection of programs which
    - provide management of databases
    - control access to data
    - contain a query language to retrieve  information easily

# Database Management System

- Manages very large amounts of data.
- Supports efficient access to very large amounts of data.
- Supports concurrent access to very large amounts of data.
  - Example: bank and its ATM machines.
- Supports secure, atomic access to very large amounts of data
  - Contrast two people editing the same UNIX file – last to write "wins" – with the problem if two people deduct money from the same account via ATM machines at the same time – new balance is wrong whichever writes last.

# Database Design

It is important to design the database in such a way that:

- *A specific item can be reached easily*
  - (maximum guarantee that the desired record will be reached)
- *The database can respond to the user's different questions easily*
  - (necessary relationships are provided)
- *The database occupies minimum storage space*
  - (choosing data types and how to express a certain concept is important)
- *The database contains no unnecessary data*
  - (storing the gross salary is enough, the net salary can be calculated from the gross salary)
- *Data can be added and updated easily without causing mistakes*
  - (no data redundancy)

# Data redundancy

- Different and conflicting versions of the same data

e.g. Employee database:

| personal info | payroll |
|---|---|
| - ID | - ID (relating parameter) |
| - name | - name (causes redundancy) |
| - address | - gross salary |

≠

# STEPS IN DATABASE DESIGN

- Requirement analysis
  - What does the user want?
- Conceptual database design
  - Defining the entities and attributes, and the relationships between these --> The ER model
- Physical database design
  - Implementation of the conceptual design using a Database Management System

# Terminology

- **Entity** --> What is this table about? students
- **Attribute** (Field) --> What items of information are necessary to keep concerning this entity?

```
ID, name, department, year, advisor
```

- **Record** (Tuple) --> A set of values for each attribute for one item

```
20091001, Student Name, CS, 1, Florin Pop
```

- **Key** --> The attribute used to define a required item who is the advisor of Student Name?
  - Types of keys:
    - *Primary Key*: Key used to uniquely identify a record
    - *Foreign Key*: A field in this table which is the Primary key of another table
- **Relationship** --> Definitions linking two or more tables

# The DBMS Marketplace

- Relational DBMS companies – Oracle, Sybase – are among the largest software companies in the world.
- IBM offers its relational DB2 system. With IMS, a nonrelational system, IBM is by some accounts the largest DBMS vendor in the world.
- Microsoft offers SQL-Server, plus Microsoft Access for the cheap DBMS on the desktop, answered by “lite” systems from other competitors.
- Relational companies also challenged by “object-oriented DB” companies.
- But countered with “object-relational” systems, which retain the relational core while allowing type extension as in OO systems.

# Query Languages

| Employee | |
|---|---|
| Name | Dept |

| Department | |
|---|---|
| Dept | Manager |

SQL

```
SELECT Manager
FROM Employee, Department
WHERE Employee.name = "Clark Kent”
        AND Employee.Dept = Department.Dept
```

Query Language

Data definition language (DDL) ~ like type defs in C or Pascal

Data Manipulation Language (DML)

Query (SELECT)

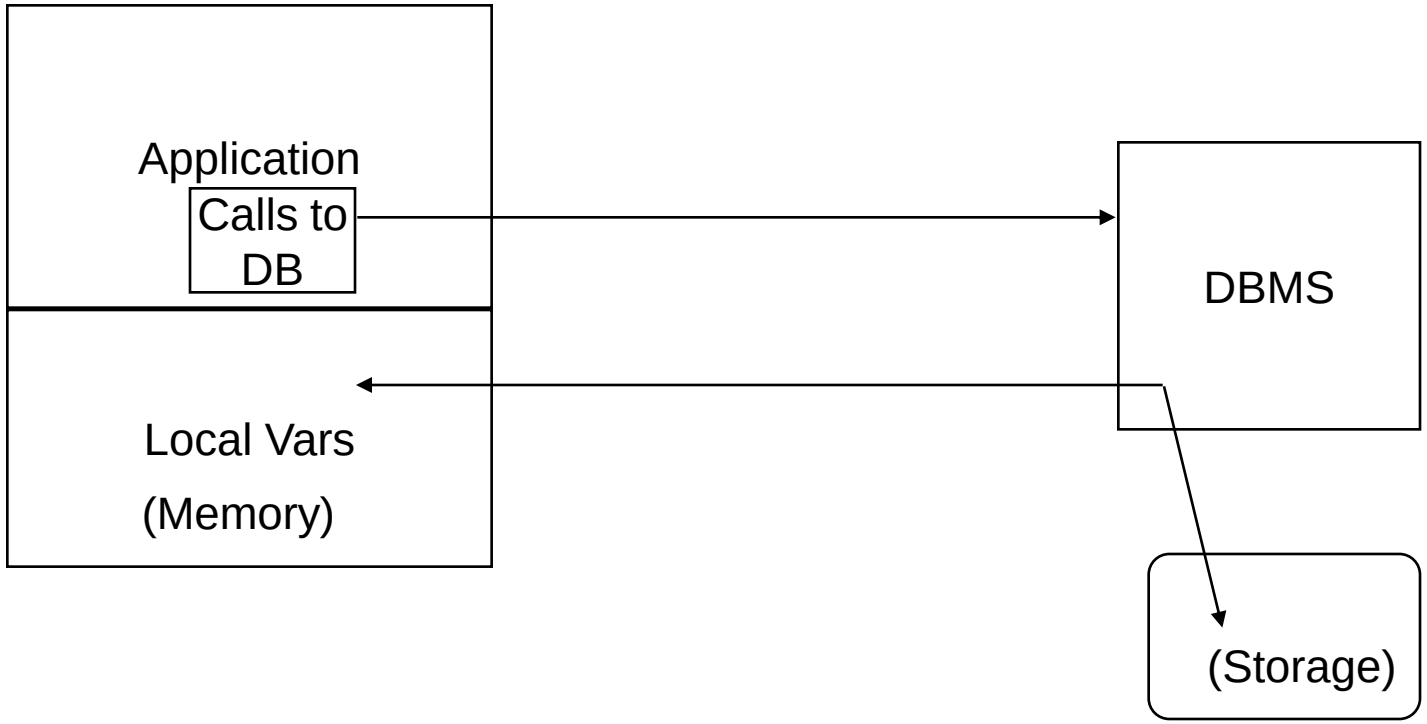
UPDATE < relation name >

SET <attribute> = < new-value>

WHERE <condition>

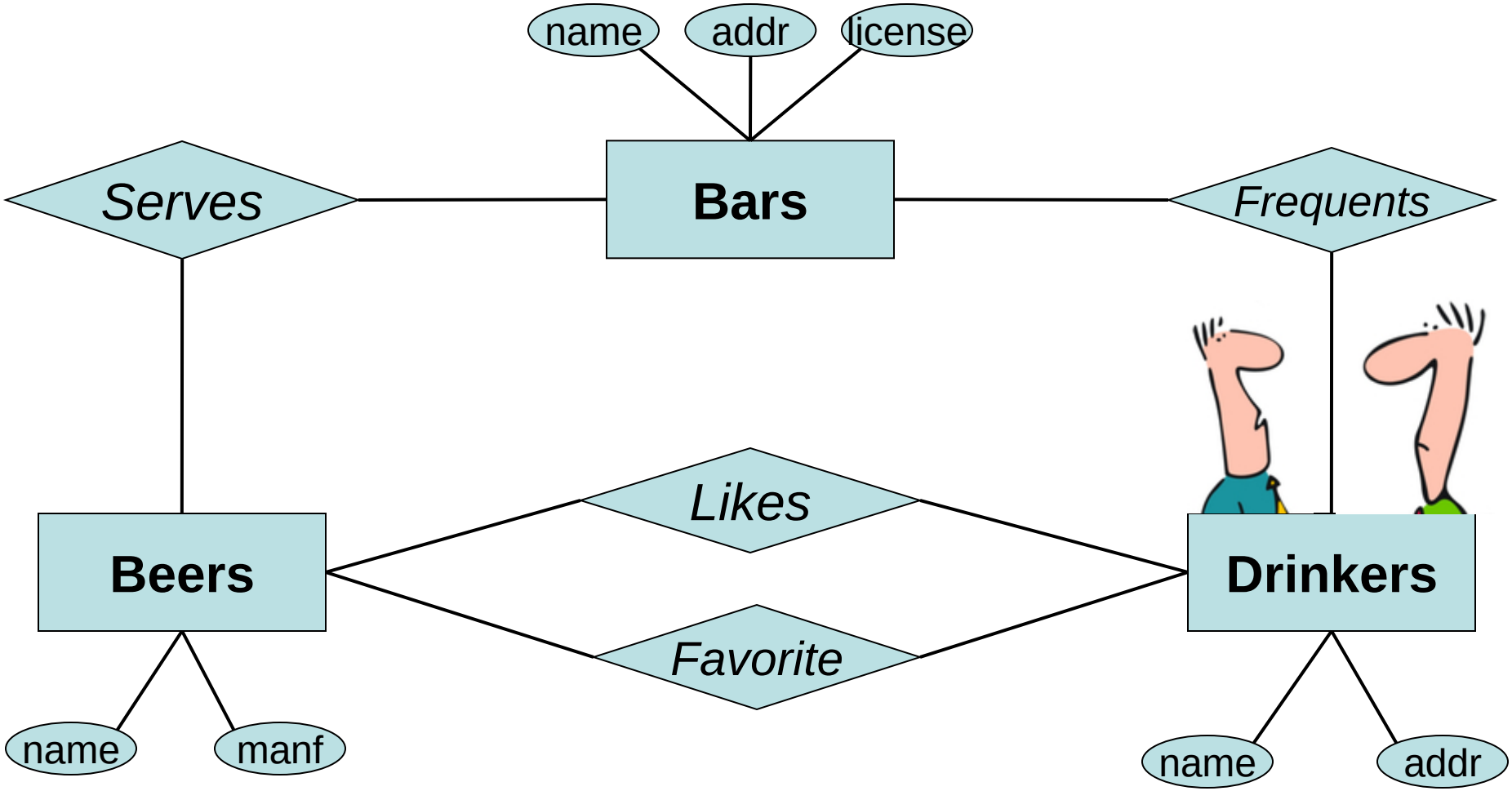
# Host Languages

C, C++, Java, Fortran, Lisp, COBOL




- Host language is completely general (Turing complete) but gives you no support
- Query language less general "non procedural" and optimizable

# Entity/Relationship Model



# Functions of a DBMS

- Indexing
- Views
- Security
- Integrity
- Concurrency
- Backup/Recovery
- Design
- Documentation
- Update/Query

# Views

- Custom representations of a database that correspond to the needs of a class of users. Stored SELECT statements.

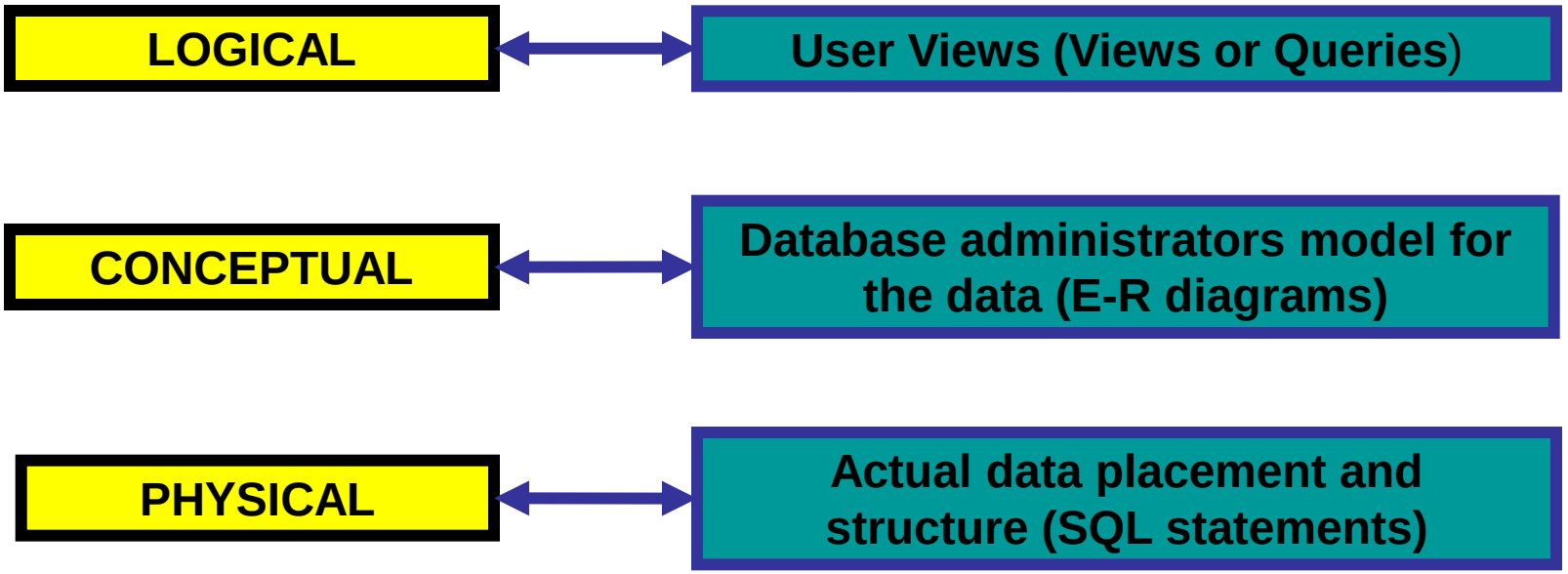
- **Views Provide:** ***representations of data for different users to***
  - ***protect data quality***
  - ***insulate users from changes in structure***

```
CREATE VIEW
        VIEWNAME {VIEW ATTRIBUTES}
        AS (SELECT ..WHERE ..)
```

# Views

**Views Permit**

- Maintaining a constant user interface
- Restricting access to specified attributes
- Specifying user rights



# Security

- Components that limit access or actions to limit potential damage to data.

- *Limit data access to properly authorized users or programs. Protect data against accidental or intentional damage.*
  - **Deter**
  - **Detect**
  - **Minimize**
  - **Recover**
  - **Investigate**

# Security Approaches

- **Views** *limit access and actions*
- **Authorization Rules** *identify users and restrict actions*
- **User Defined Procedures** *in addition to database security functions*
- **Encryption** *encode stored data*
- **Authentication** *positively identify users*

# User Defined Procedures

***Code modules that enforce security procedures are run during processing***

# Authorization Rules

| Subject | Object | Action | Constraint |
|---|---|---|---|
| Sales Dept | Cust | Insert | Credit < $5000 |
| Program Ar4 | Order | Modify | None |
| Terminal 12 | Cust | Modify | Balance Due |
| Order Trans | Cust | Read | None |

- ***DBMS products authorize actions based on specific records and functional descriptions.***
- ***DBMS's limit actions on tables to one of:***
  - **Read:** view but not change
  - **Insert:** read and add records
  - **Update:** read, insert and change records
  - **Alter/Delete:** read, insert, update and delete records, change table structure

# Integrity

- Components that preserve the relationship among different related records in the database

- *The relationship among records in the database*
  - Referential Integrity
  - Non Key Integrity
  - Derived Conditions

# Constraints in SQL

`CREATE TABLE` ... or

`ALTER TABLE ... ADD`

- CHECK (*condition*)
- PRIMARY KEY *attribute-name*
- FOREIGN KEY *attribute-name* REFERENCES *parent-table*

***The parent table must already have a primary key defined***

# Concurrency

- Preventing two users from interfering with each other when they use the same information
- Lockout
  - *Restricting access to users who could be misled by partial transactions*
- Versioning
  - *Making trial updates on versions of the database and denying one if there is a data conflict.*

# Locks

- On INSERT or UPDATE statements
- SELECT *column-names*
  FROM *table-names*
  WHERE …
  FOR UPDATE OF *column-names*
  NOWAIT;

Master

| | Student | Grade |
|---|---|---|
| 00 | Fred | |
| 01 | Anthony | |
| 02 | Steve | |
| 03 | Ivan | |

**Program 1 locks record <exclusive>.**

*No other program can read the record.*
*No program can have an active lock.*

**Program 2 locks record <shared>**

*Other programs can read, but not change record.*
*No program can have an exclusive lock.*

# Concurrency - Locks

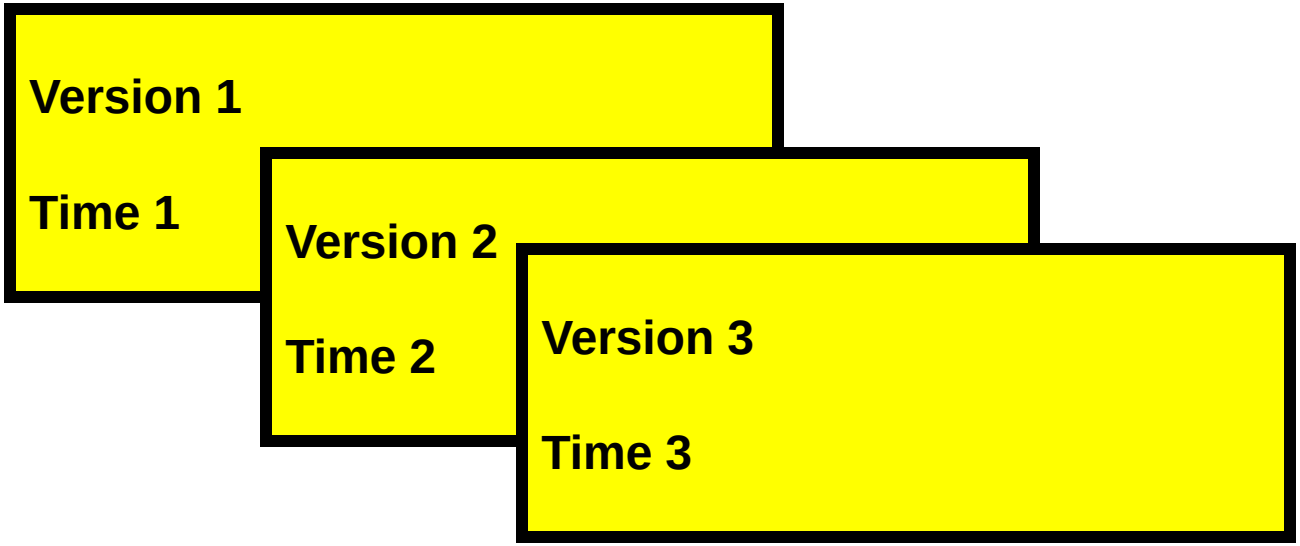
- **Granularity**
  - Field
  - Record
  - Table
  - Database
- **Exclusivity**
  - Exclusive
  - Shared

# Concurrency - Deadlock

- Two programs request conflicting sets of data lock up the database while awaiting access.
  - Program 1 locks record A
  - Program 2 locks record B
  - Program 1 requests  lock on record B; waits
  - Program 2 requests  lock on record A; waits

- System either times out and restarts each transaction after a random wait or recognizes the deadlock to abort one program.

# Versioning




**Commits version 3 only after changes to versions 1 and 2 have been rolled back.**

# Backup and Recovery

- Processes to confirm and repeat transactions so that database can be restored to a valid state after a problem.

- Backup Copies
  - Master
  - Transaction Log
- Journalization
  - Forward Log
  - Backward Log
- Checkpoints

# DBMS Logs

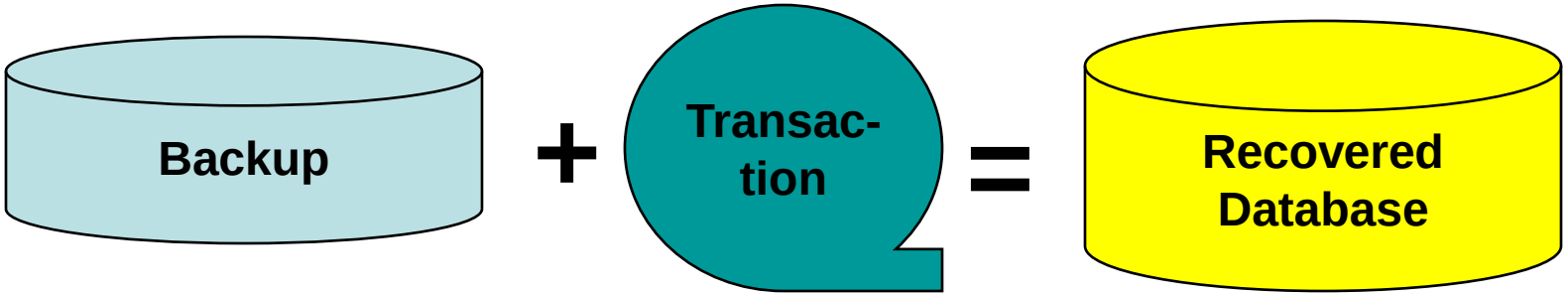
Master

| | Student | Grade |
|---|---|---|
| 00 | Fred | |
| 01 | Anthony | |
| 02 | Steve | |
| 03 | Ivan | |

Transaction

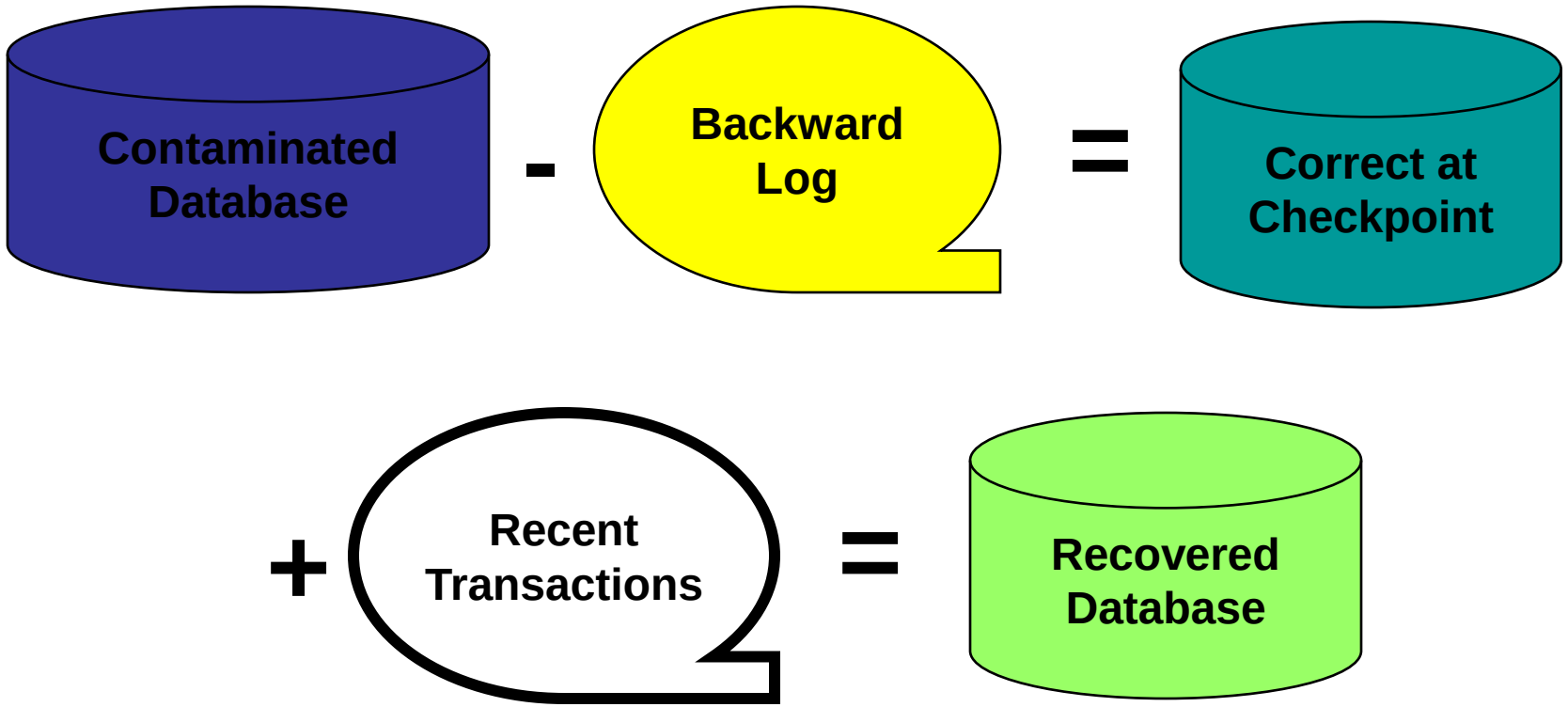
| |
|---|
| Insert Li with grade A |
| Change Fred’s grade to A |
| |

# Recover from Backup




- Slow
- May give different answers from original

# Recover to Checkpoint Using Logs



# Transaction Processing

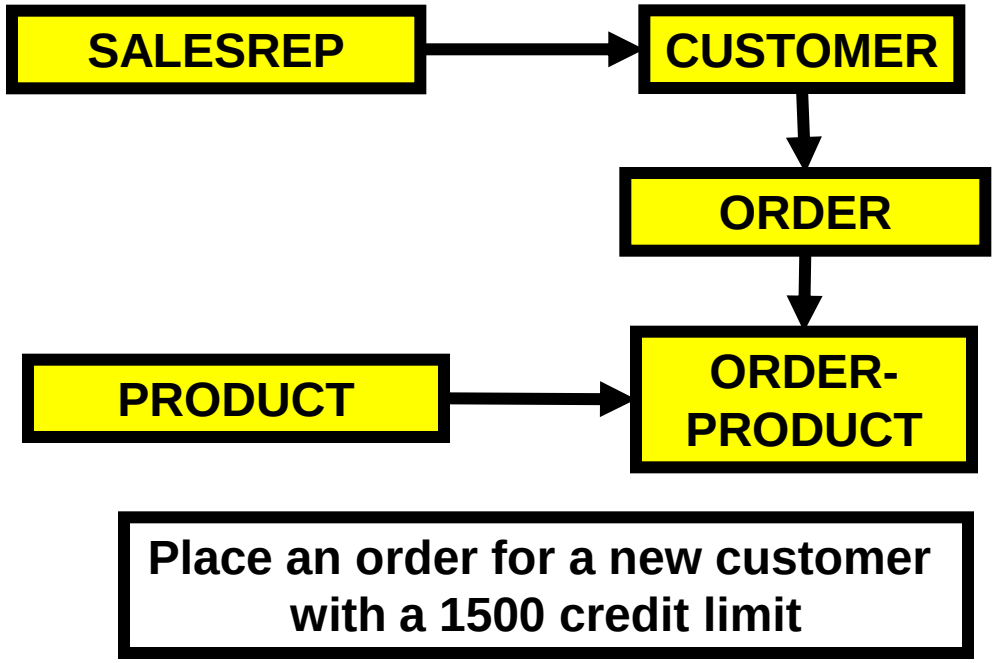
- *A set of computer operations required to process a single unit of work.*
- A transaction must conclude with the database in a valid state whether the transaction terminates correctly or abnormally

- Transaction Boundary
  - Locking
    *Exclusive* *Shared*
  - Logging
    *Forward* *Backward* *Transaction*
  - Modification
    *Delete* *Insert* *Update*
  - Commitment
    *Commit* *Rollback*

# Transaction Boundaries

- Set Boundary
  - Obtain Locks
  - Execute Code Modules
  - Evaluate Correctness
- Commit or Rollback
  - Release Locks

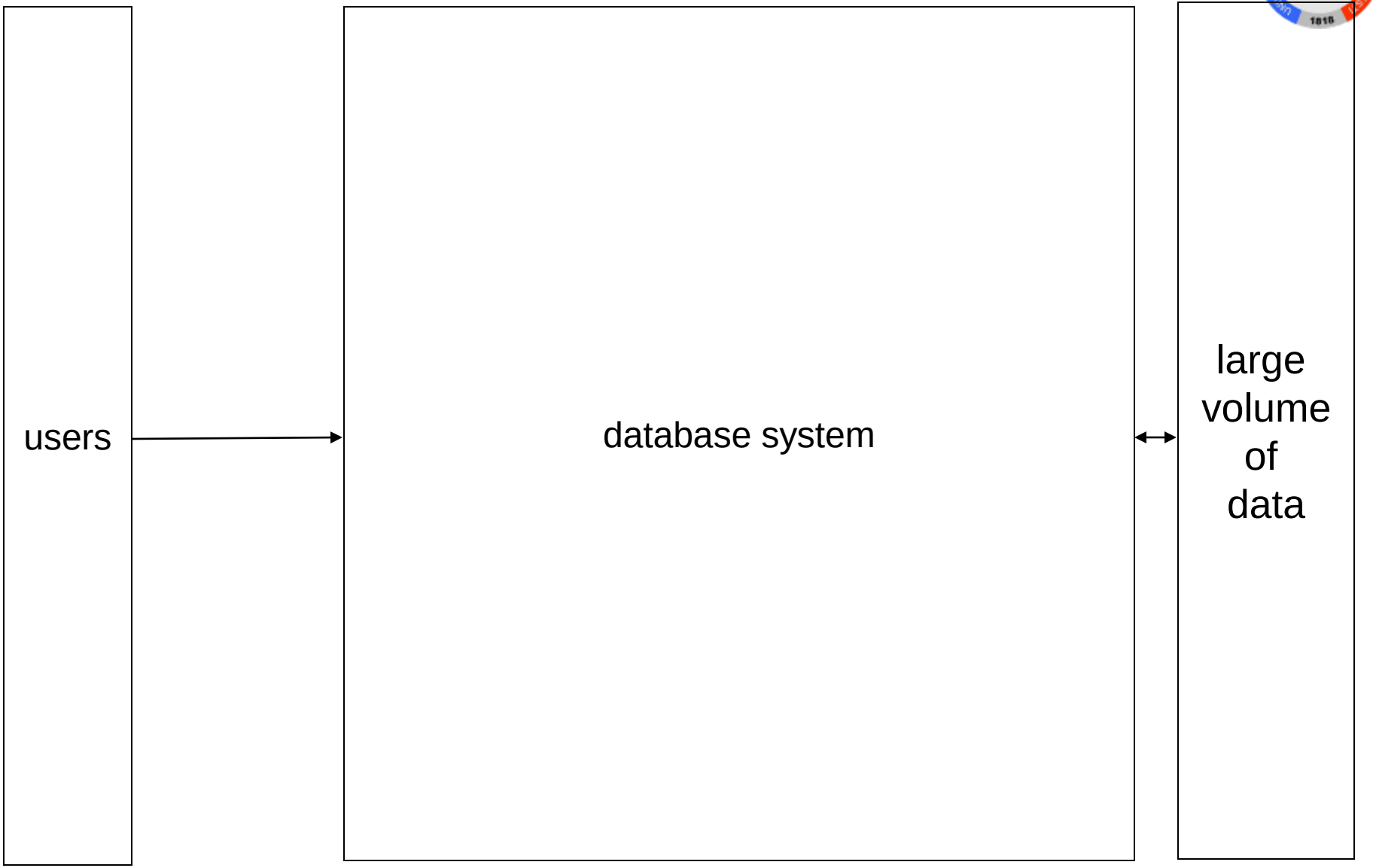
- Set savepoint:
  SAVEPOINT order_save;
- Commit or rollback:
  ROLLBACK TO order_save;

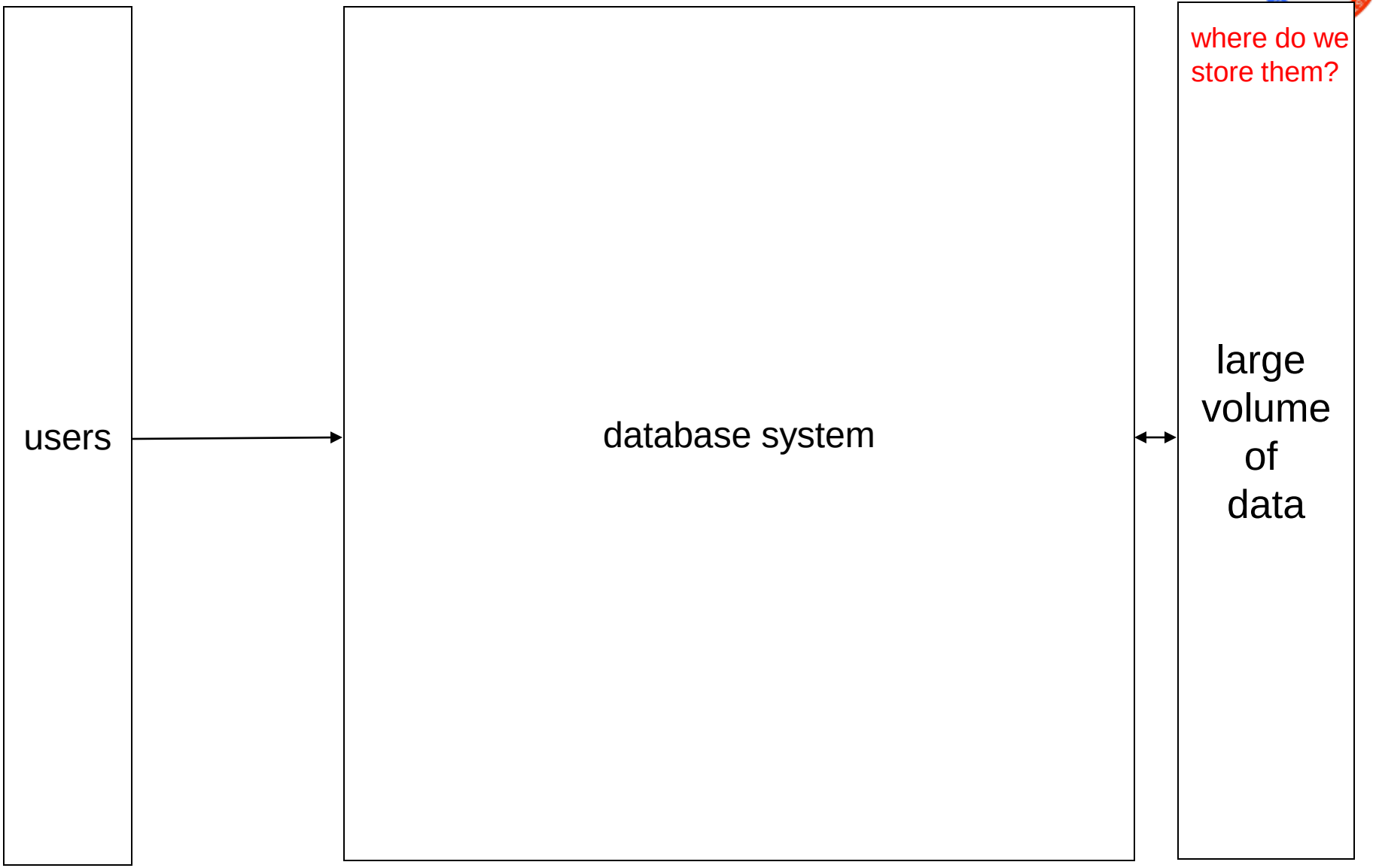


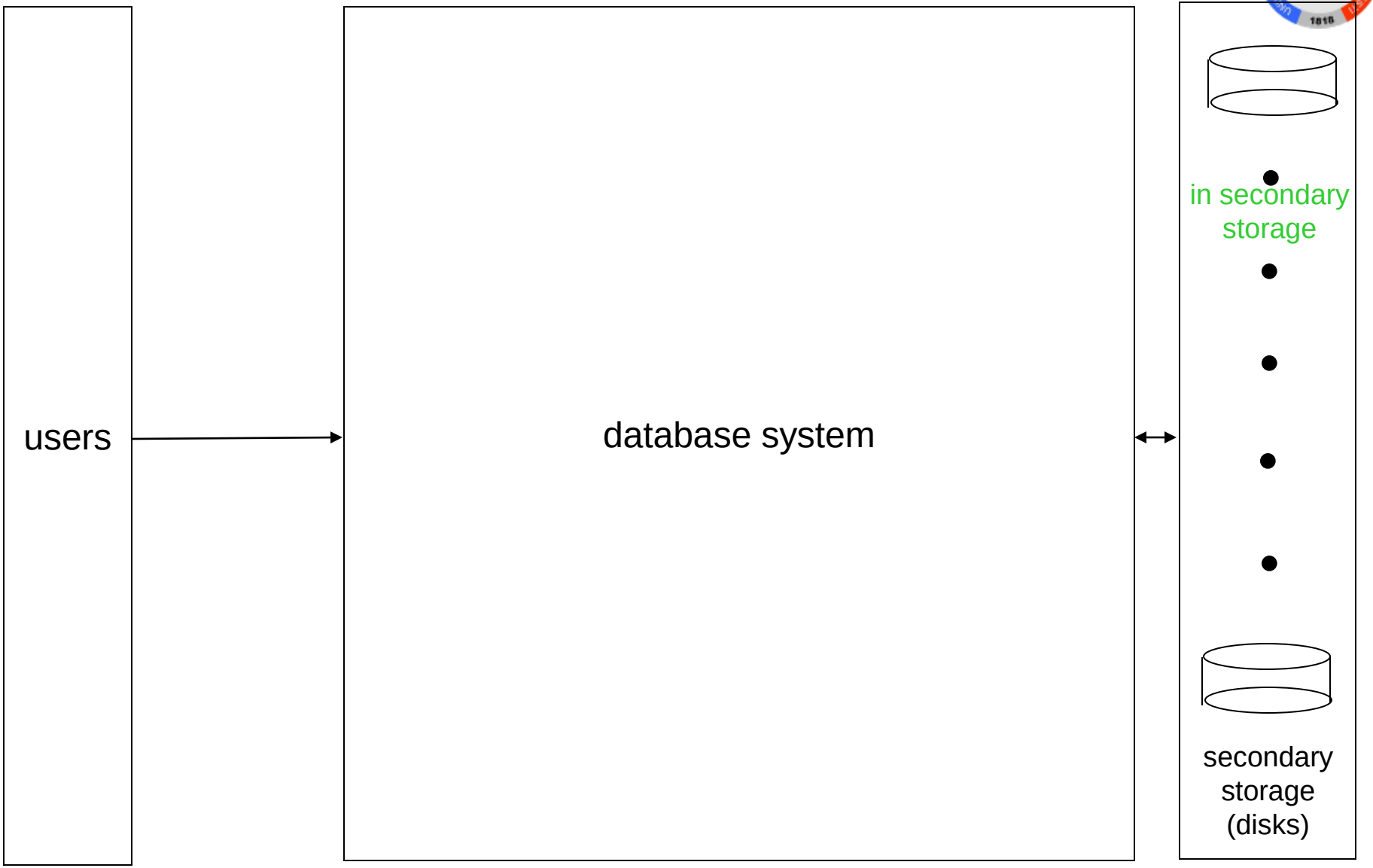

- **Insert CUSTOMER Record**
- **Update CUSTOMER with SALESREP For. Key**
- **Insert ORDER Record**
- **Insert ORDER-PRODUCT with Foreign Keys**
- **Update ProductOnHand in PRODUCT**
- **Check Credit Limit**

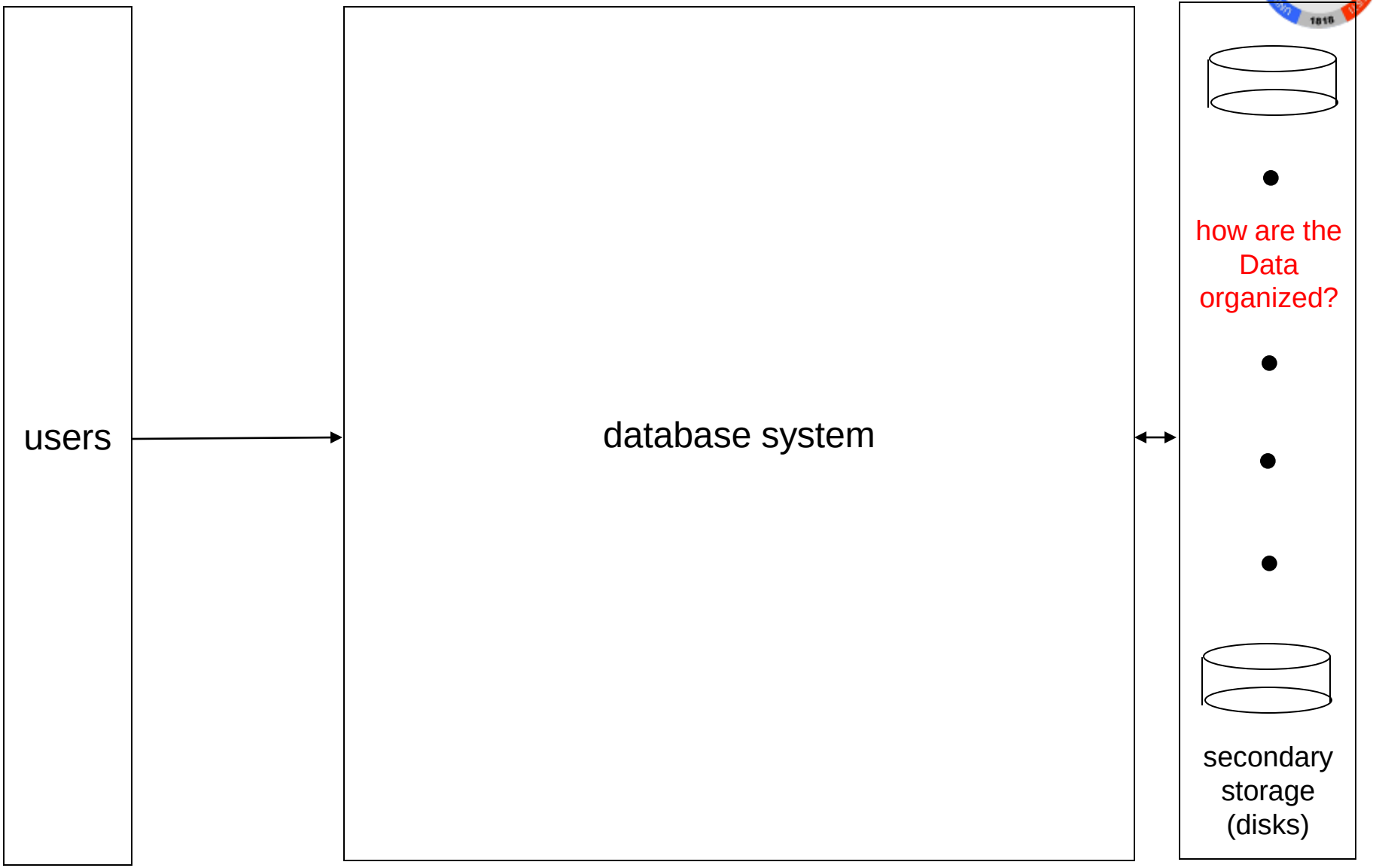
# Transaction Processing Programming Logic

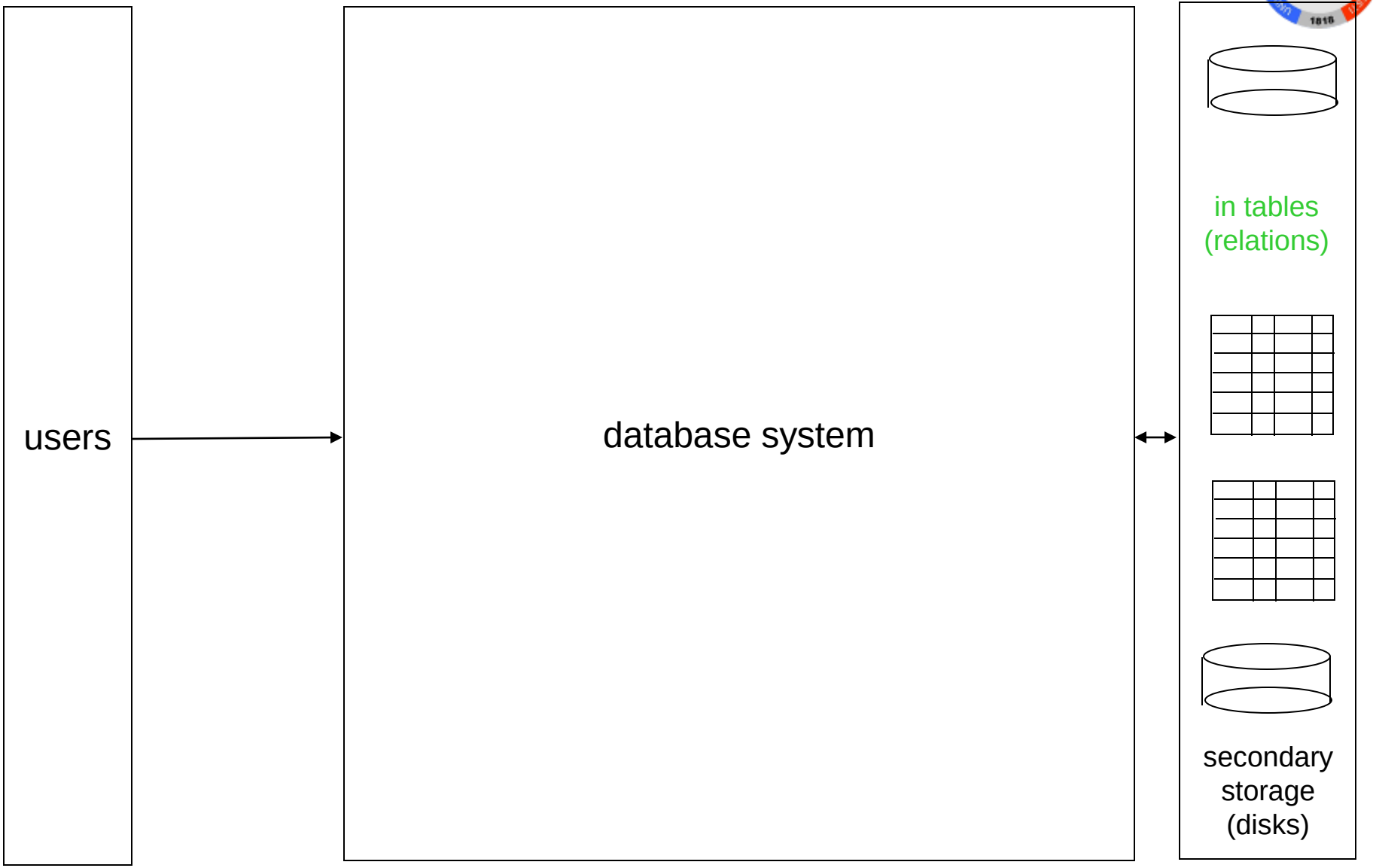
- *Two phased locking requires obtaining locks on all necessary records before releasing locks on any records.*
  - Obtain locks on all records needed
  - Perform calculations
  - Release locks

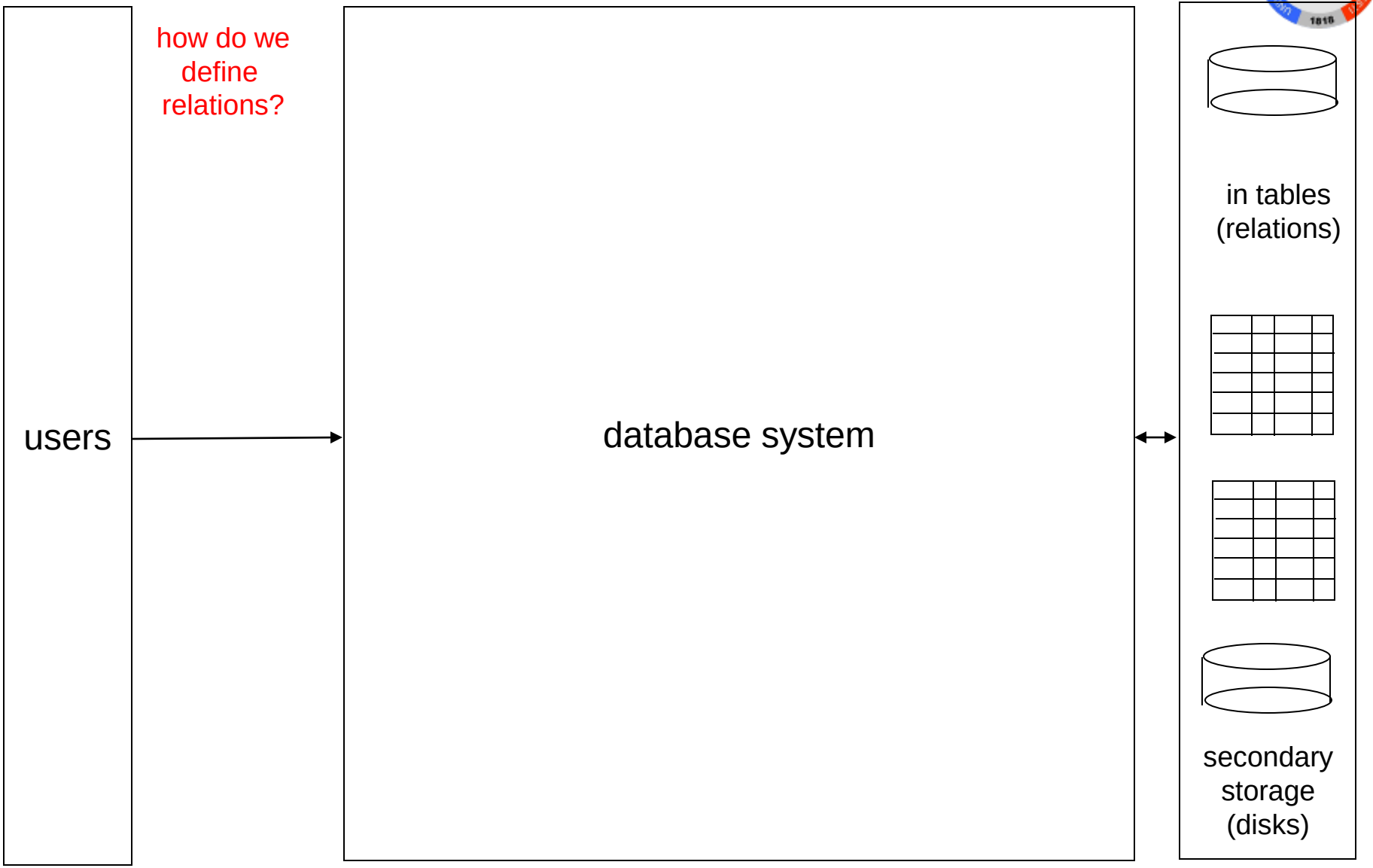

users
database system
large volume of data


users
database system
where do we store them?
large volume of data

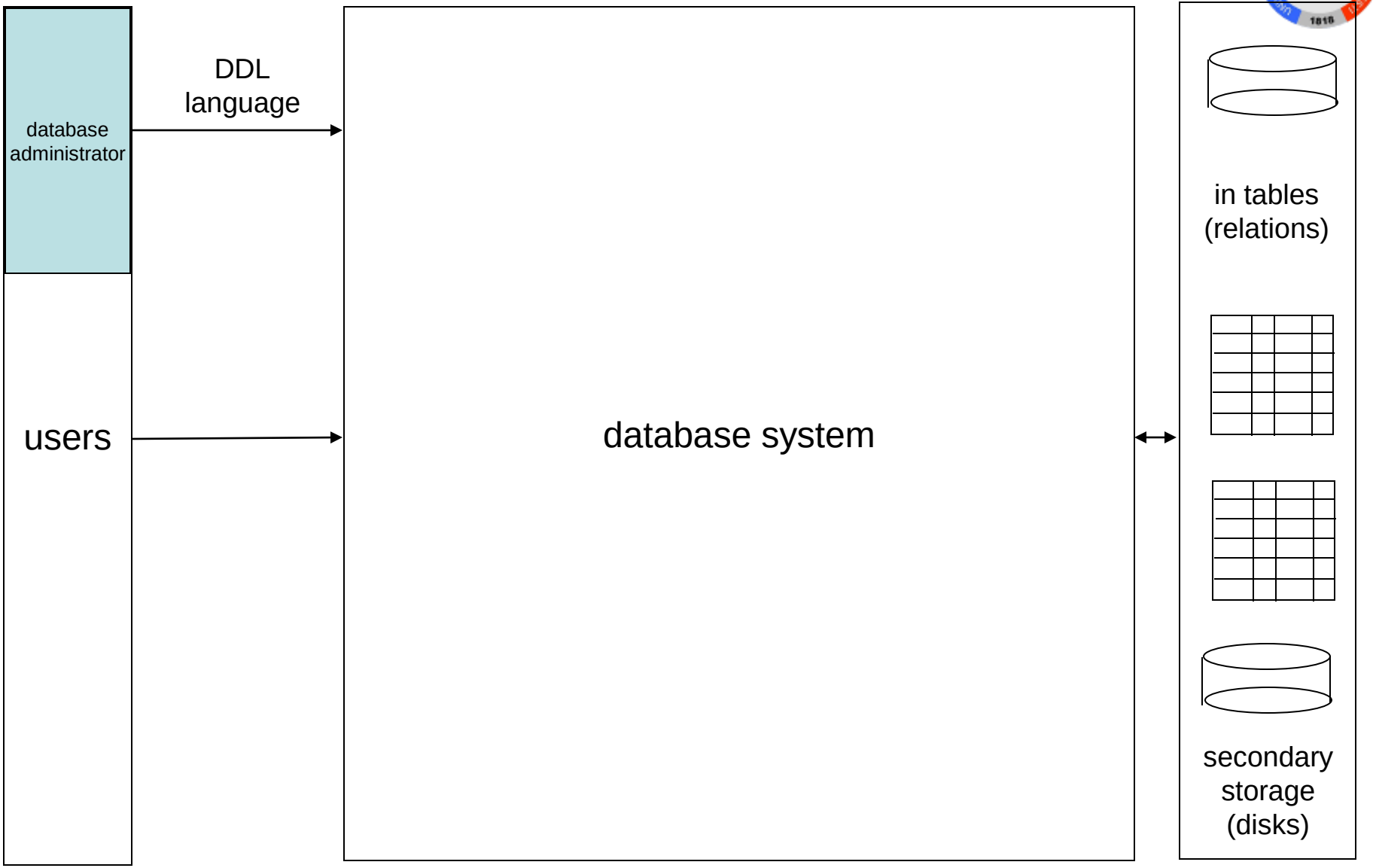

users
database system
in secondary storage
secondary storage (disks)

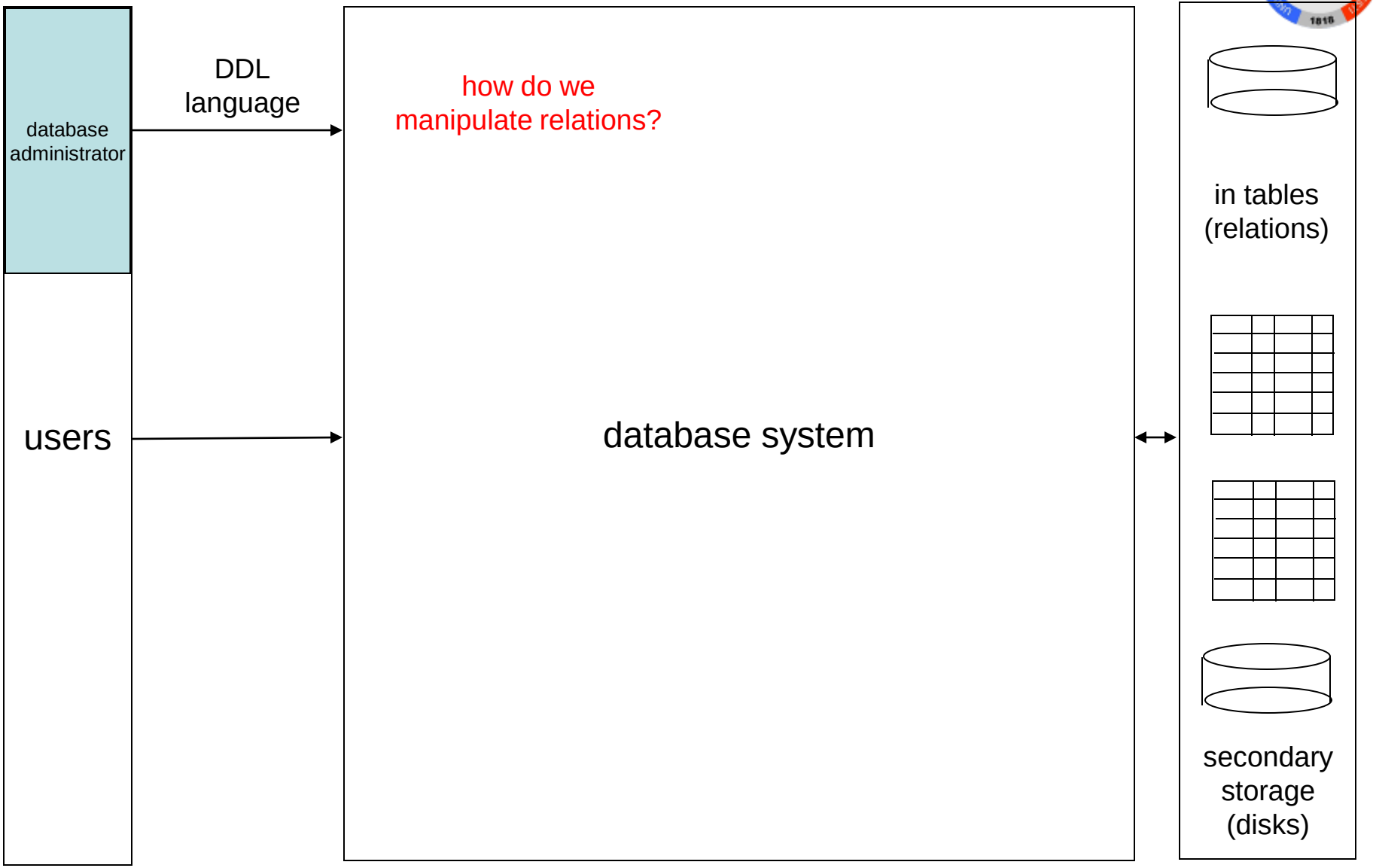

users
database system
how are the Data organized?
secondary storage (disks)


users
database system
in tables
(relations)
secondary
storage
(disks)

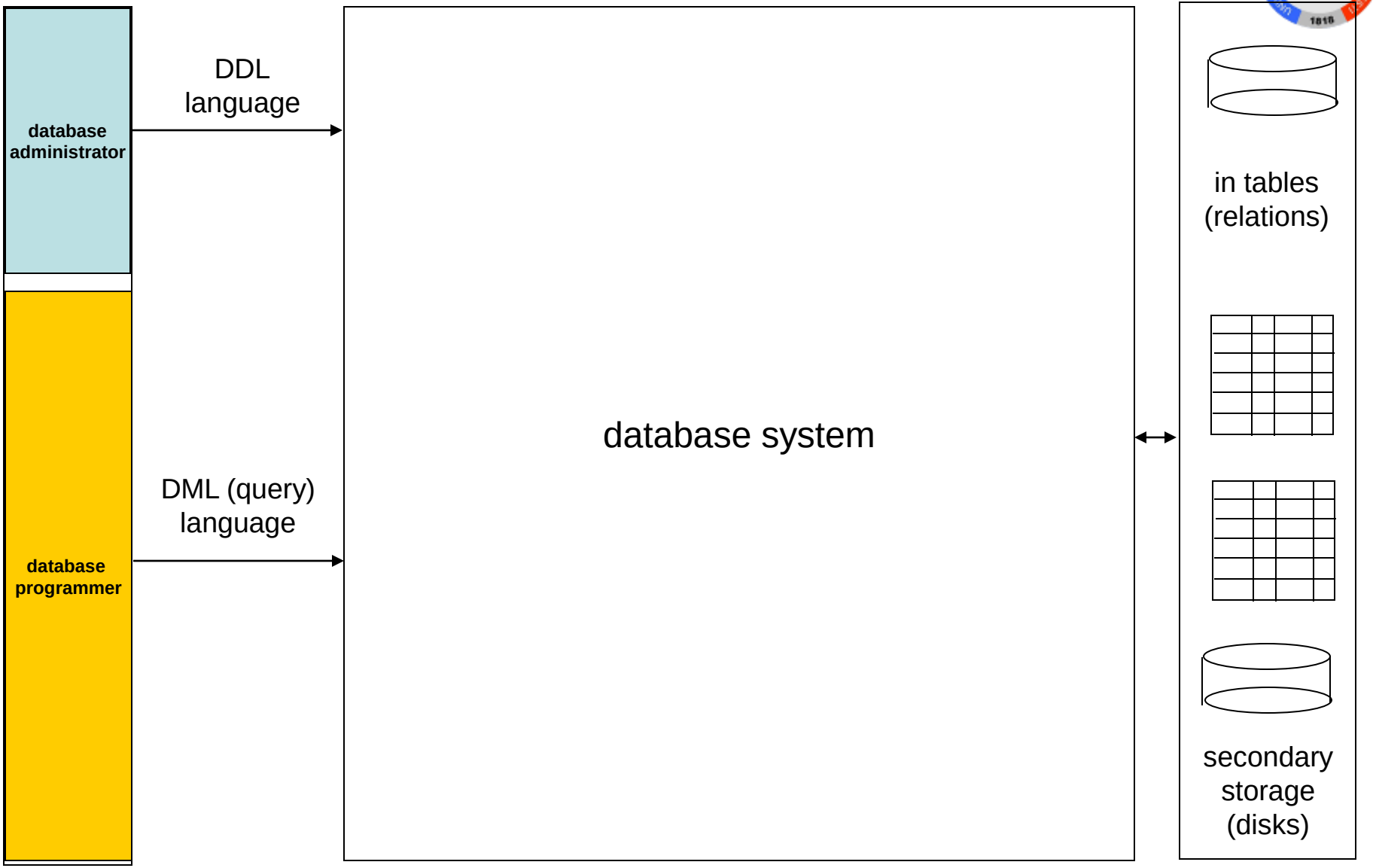

how do we define relations?
users
database system
in tables (relations)
secondary storage (disks)

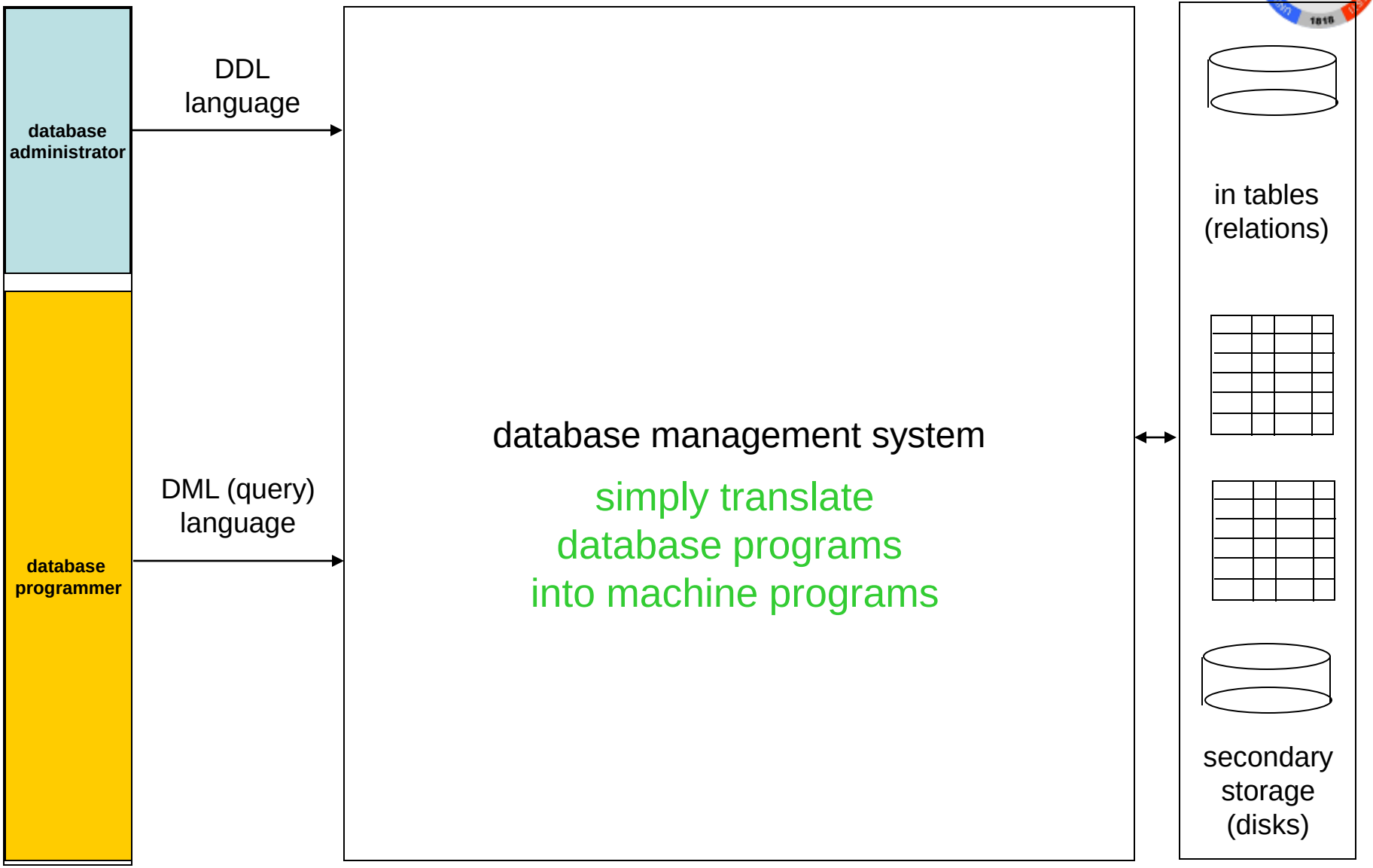
# DDL = data definition language

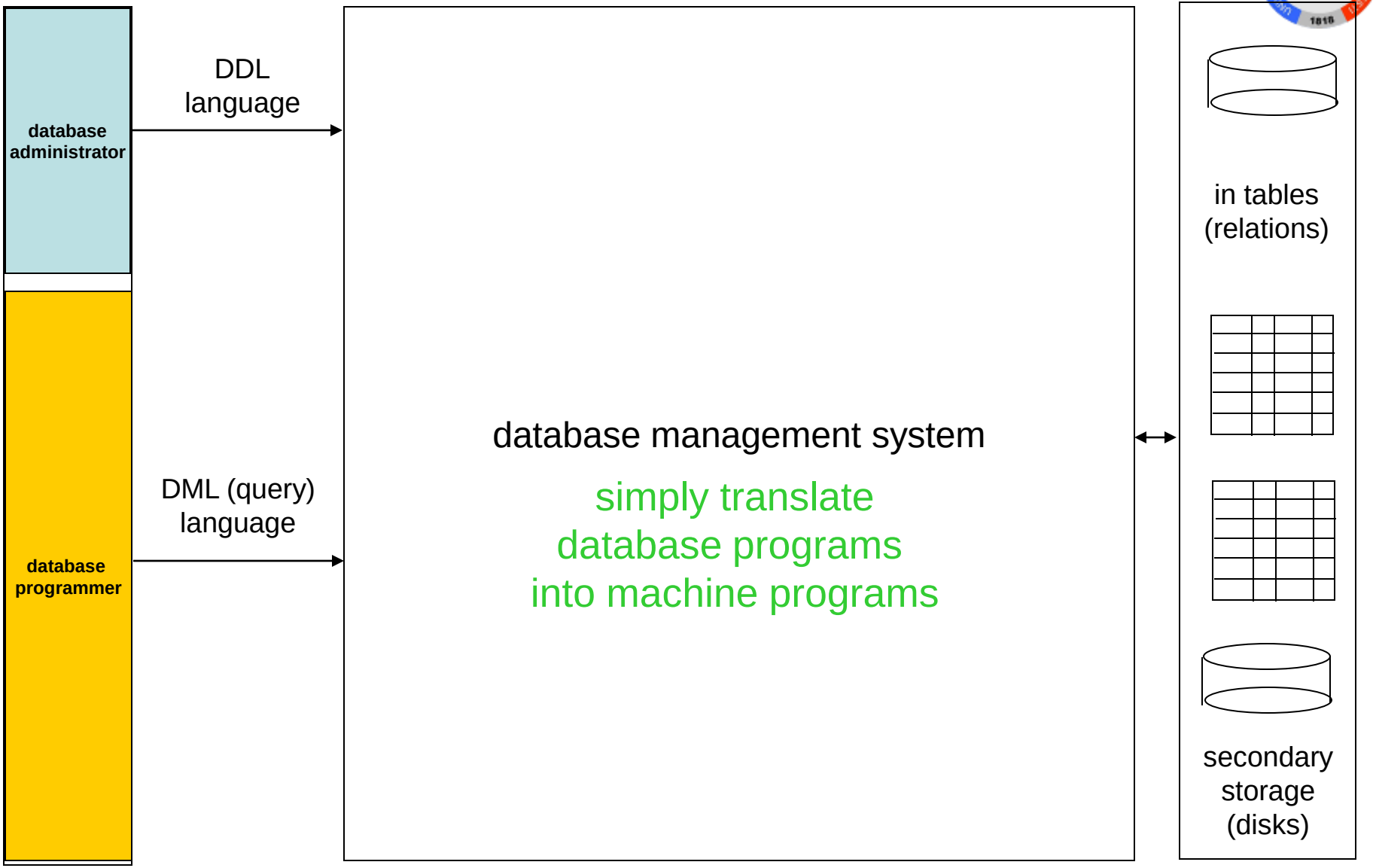



database administrator
DDL language
how do we manipulate relations?
users
database system
in tables (relations)
secondary storage (disks)

# DML = data manipulation language

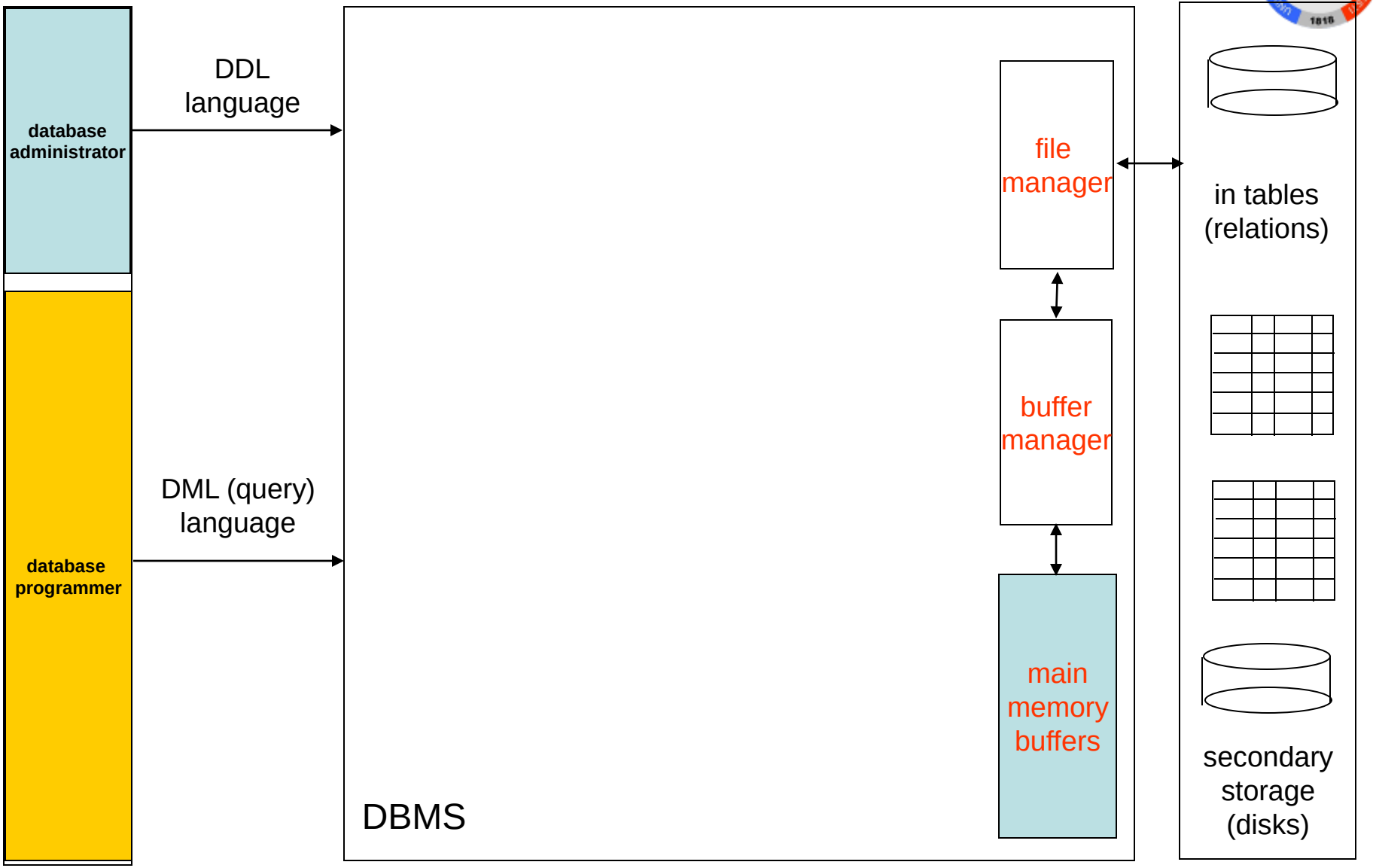



database administrator
DDL language
database programmer
DML (query) language
database management system
simply translate
database programs
into machine programs
in tables (relations)
secondary storage (disks)



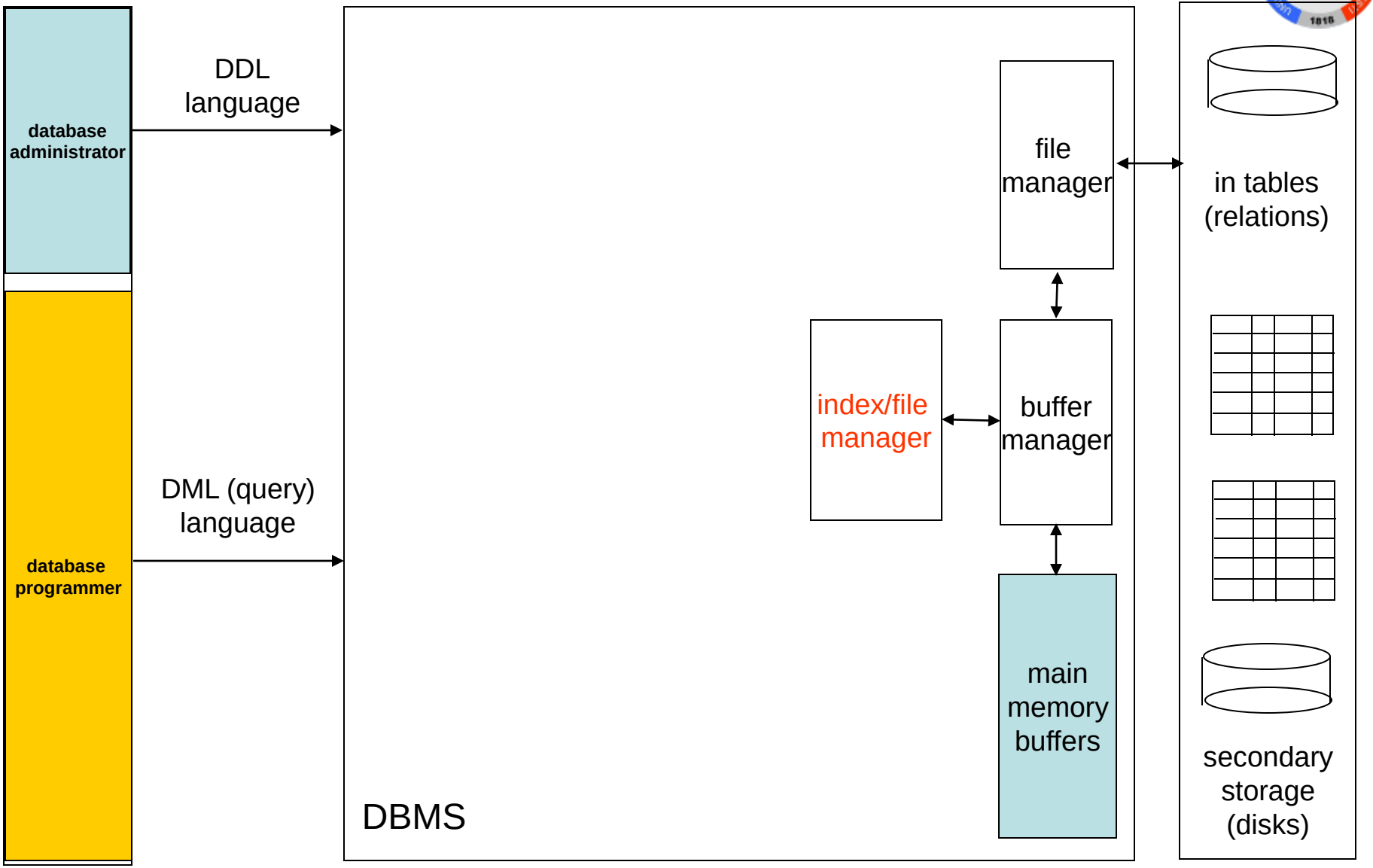

Then what is the difference between DBMS and a programming language compiler?

# 1. It has to deal with data stored in hierarchical memory structures



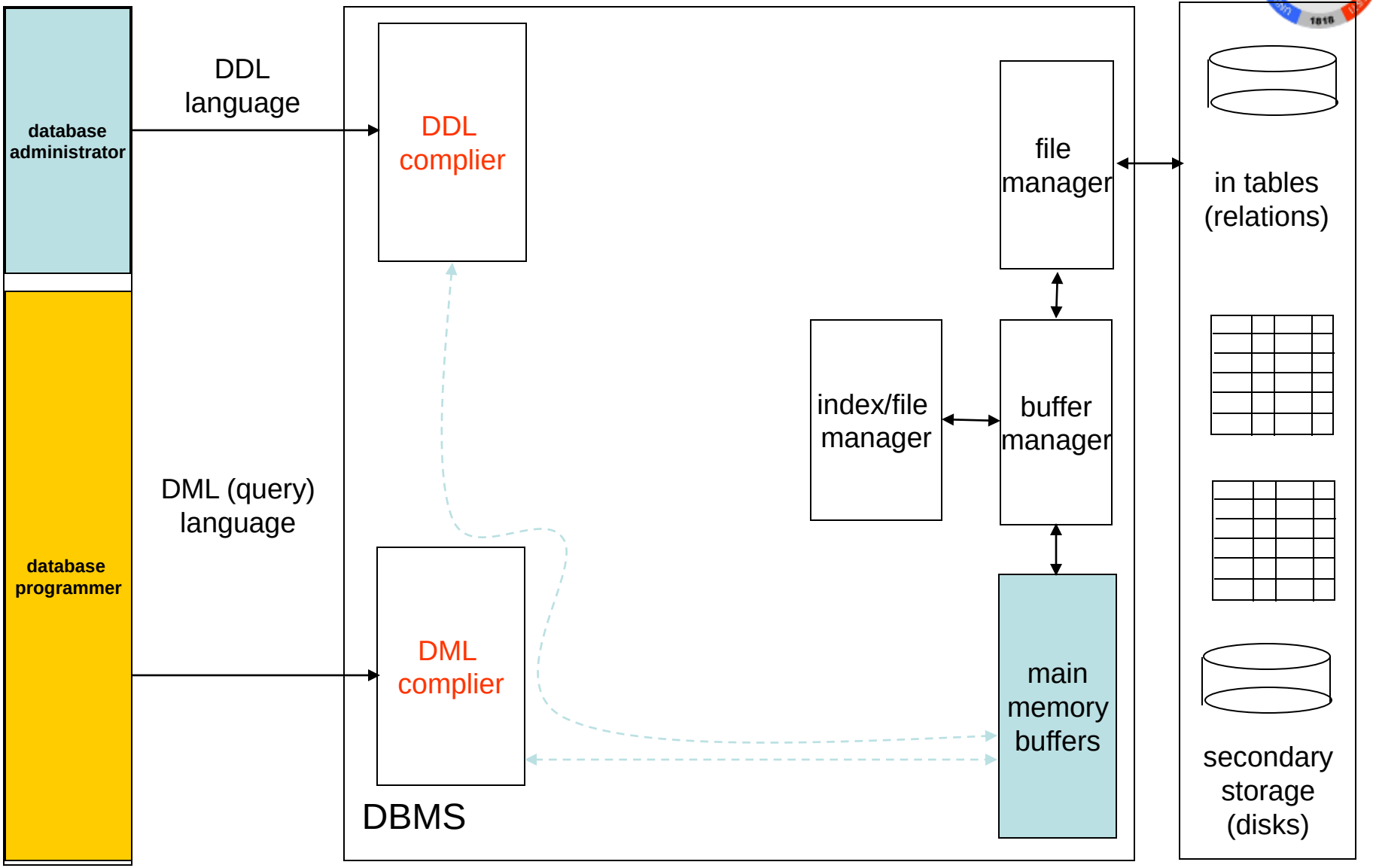

Then what is the difference between DBMS and a programming language compiler?

2. It has to support efficient manipulations of data in hierarchical memory structures



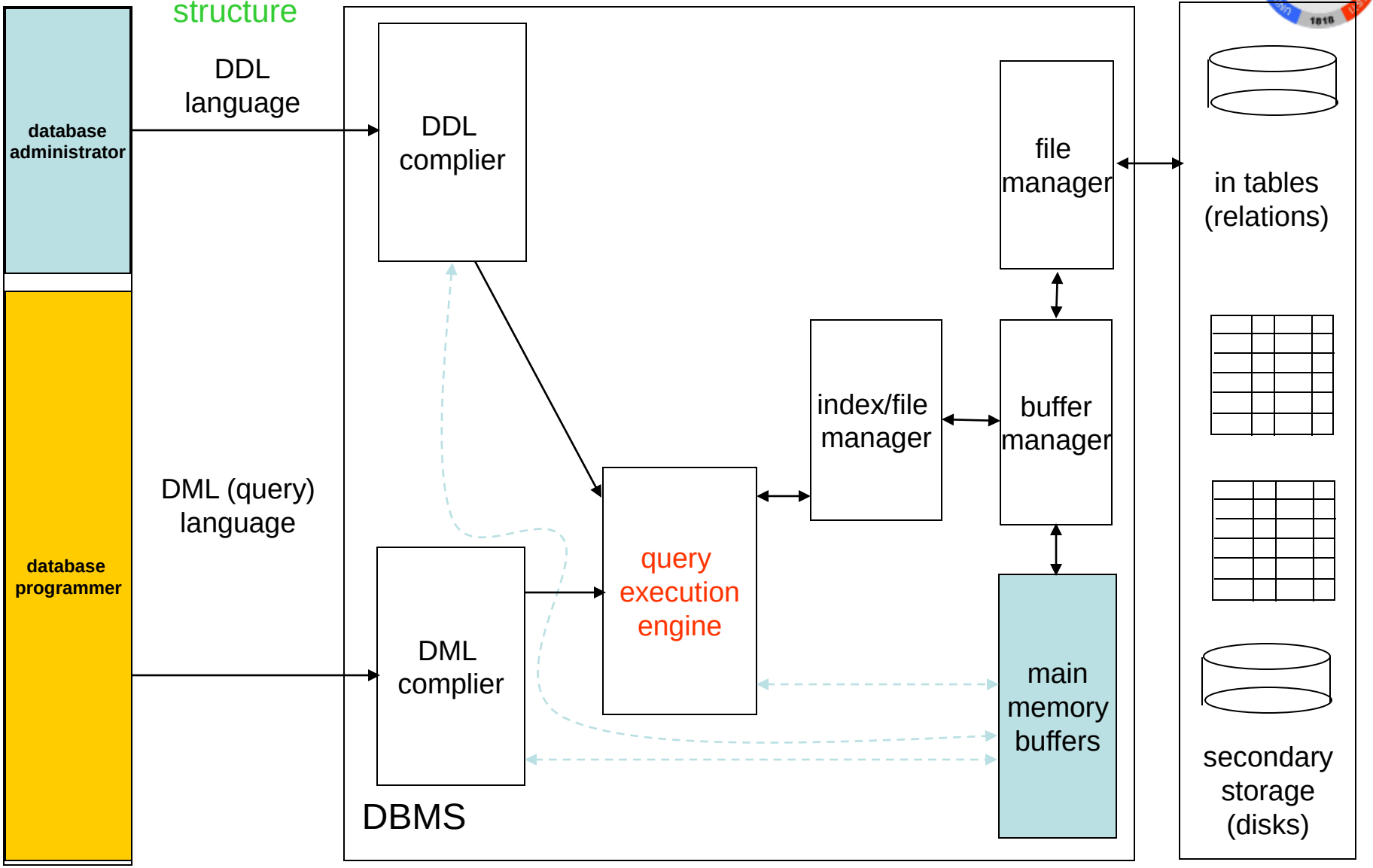

Then what is the difference between DBMS and a programming language compiler?

3. It needs to translate the input database program into an internal representation



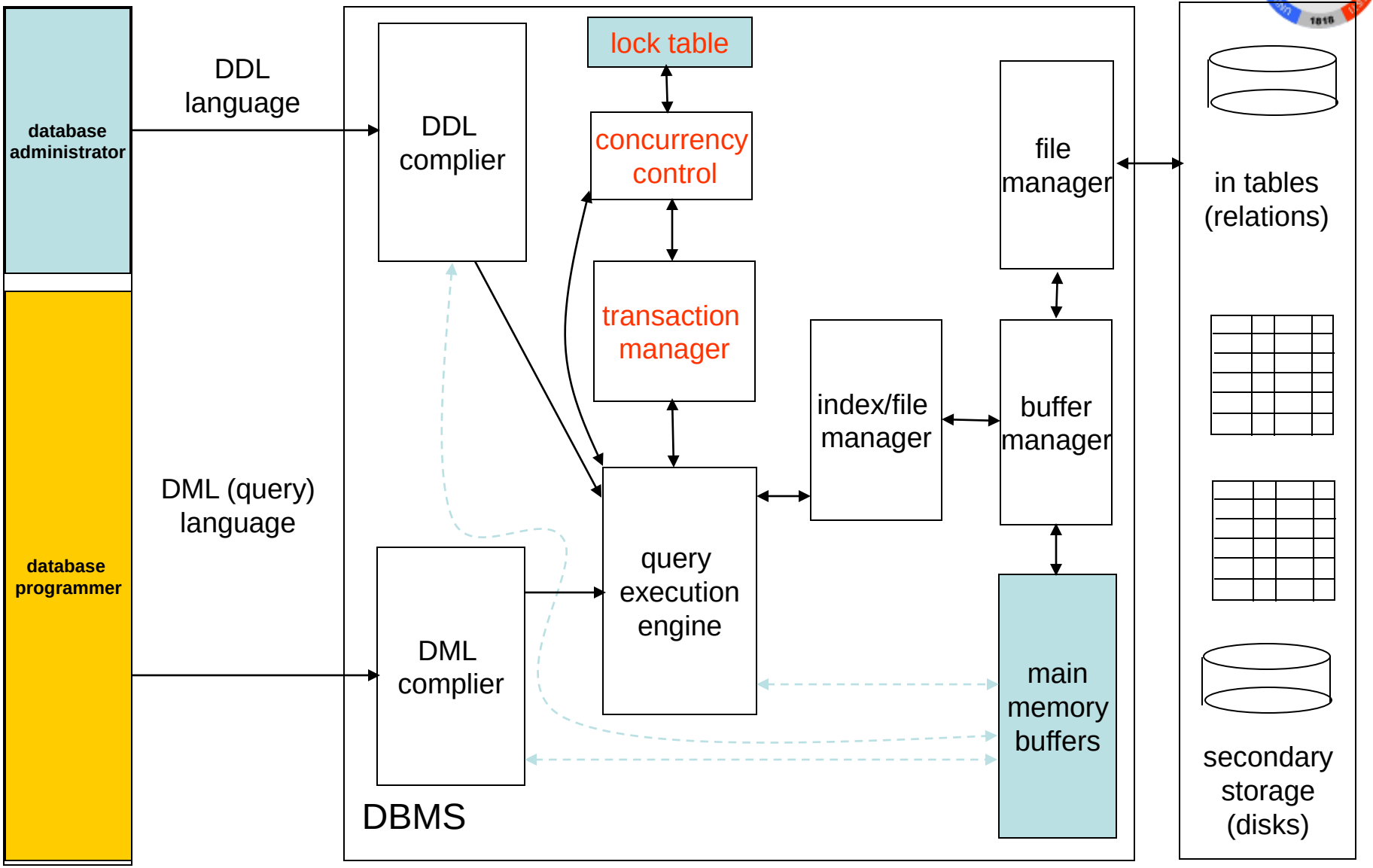

Then what is the difference between DBMS and a programming language compiler?

4. It needs to produce efficient internal codes dealing with data in hierarchical memory structure



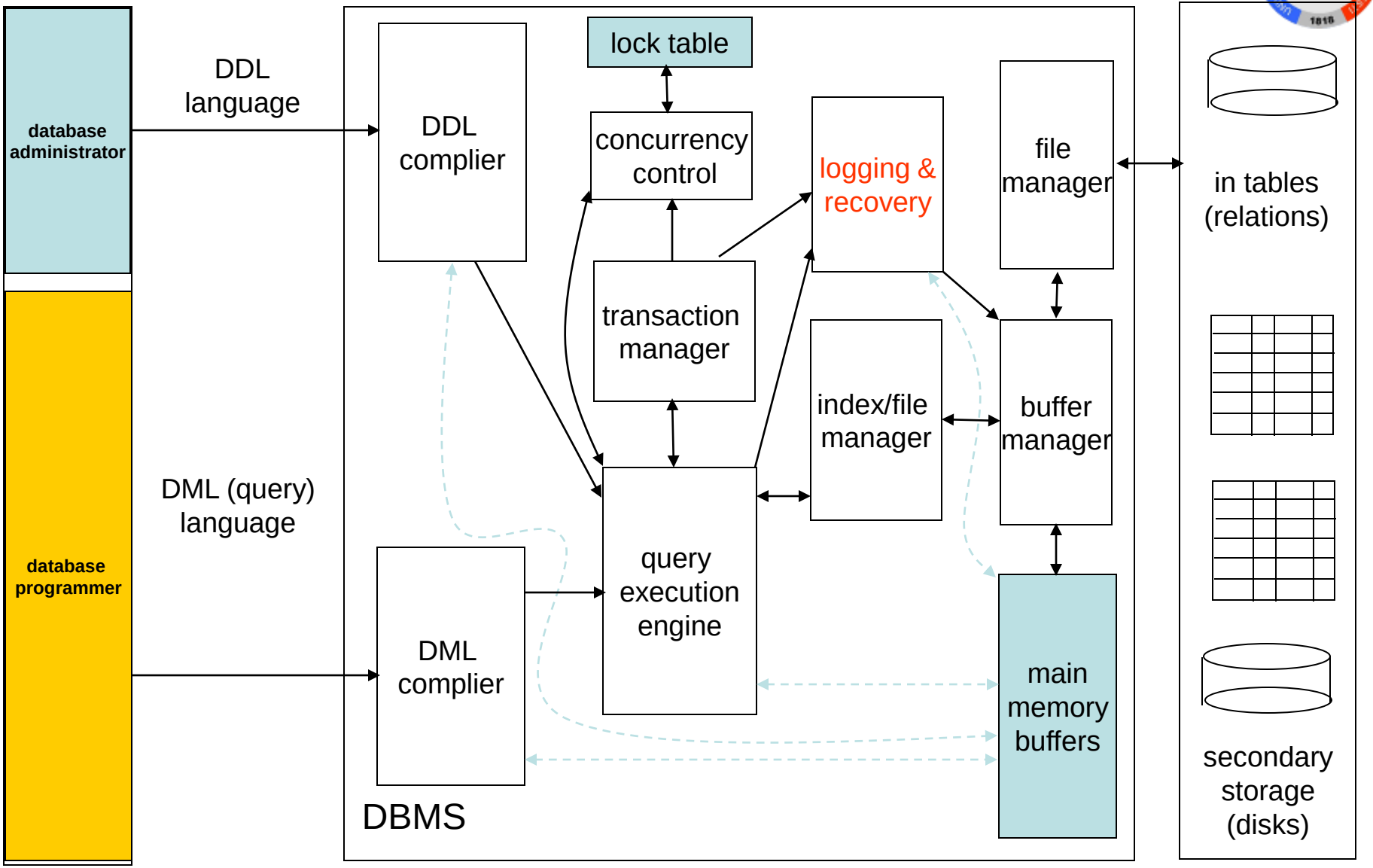

Then what is the difference between DBMS and a programming language compiler?

# 5. It needs to be consistent

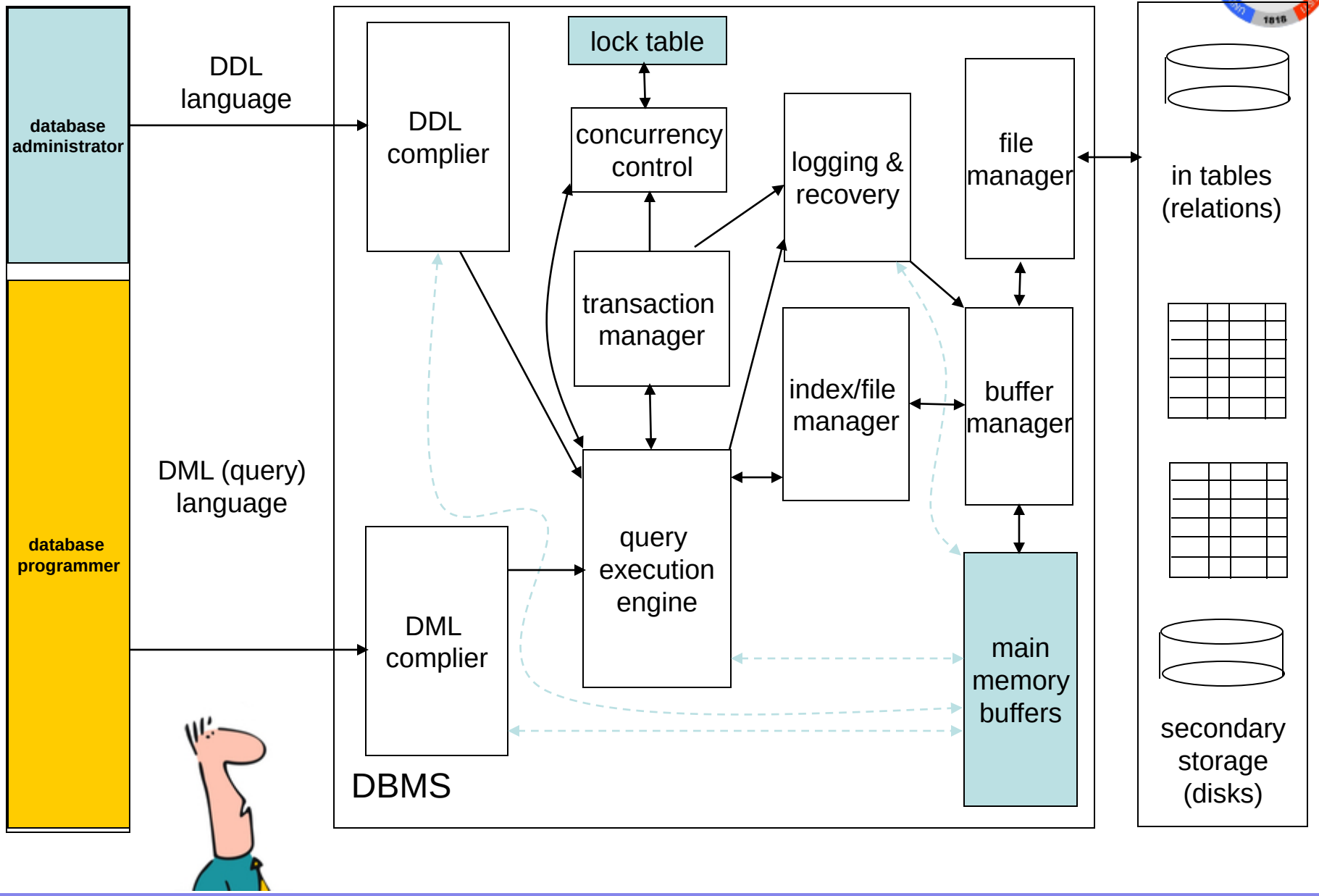



Then what is the difference between DBMS and a programming language compiler?

# 6. It needs to be reliable




Then what is the difference between DBMS and a programming language compiler?


database administrator
database programmer
DDL language
DML (query) language
DDL complier
DML complier
lock table
concurrency control
transaction manager
logging & recovery
file manager
index/file manager
buffer manager
query execution engine
main memory buffers
DBMS
in tables (relations)
secondary storage (disks)

# Exam's quizzes

- **1.** Ce este un sistem de gestiune a bazelor de date. Care este modelul frecvent folosit pentru realizarea SGBD-urilor?
- **2.** Descrieți procesul de proiectare a unei baze de date.
- **3.** Descrieți pe scurt funcțiile unui SGBD.
- **4.** Realizați o descriere sumară a arhitecturii unui SGBD.