

**NOVA**

**IMS**

Information  
Management  
School

# Storing and Retrieving Data

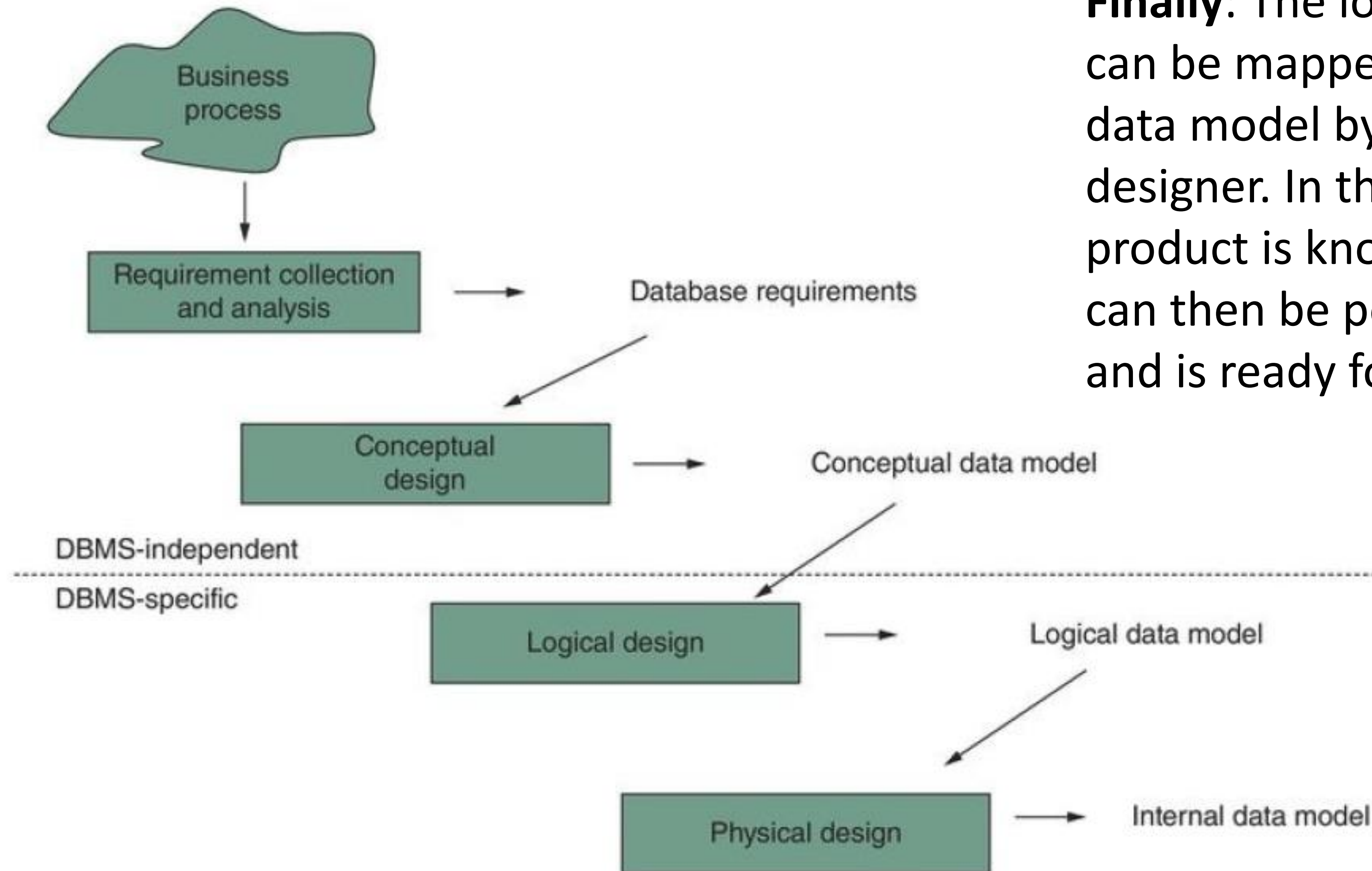
## Lecture 2

### Database modelling

Lecturer: Mijail Naranjo Zolotov  
Email: [mijail.naranjo@novaims.unl.pt](mailto:mijail.naranjo@novaims.unl.pt)

# Previous class

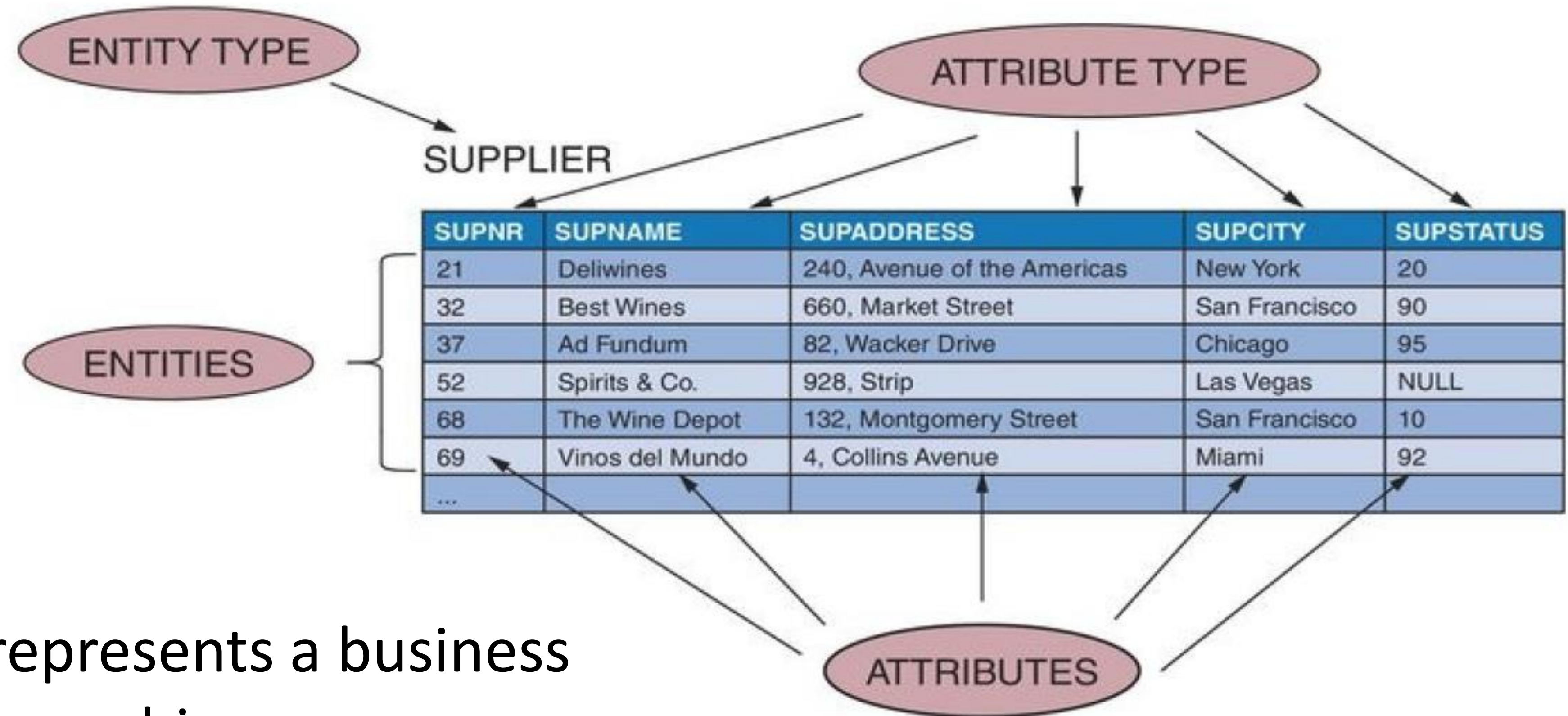
# The database design process



**Finally:** The logical data model can be mapped to an internal data model by the database designer. In this step, the DBMS product is known. The database can then be populated with data and is ready for use.

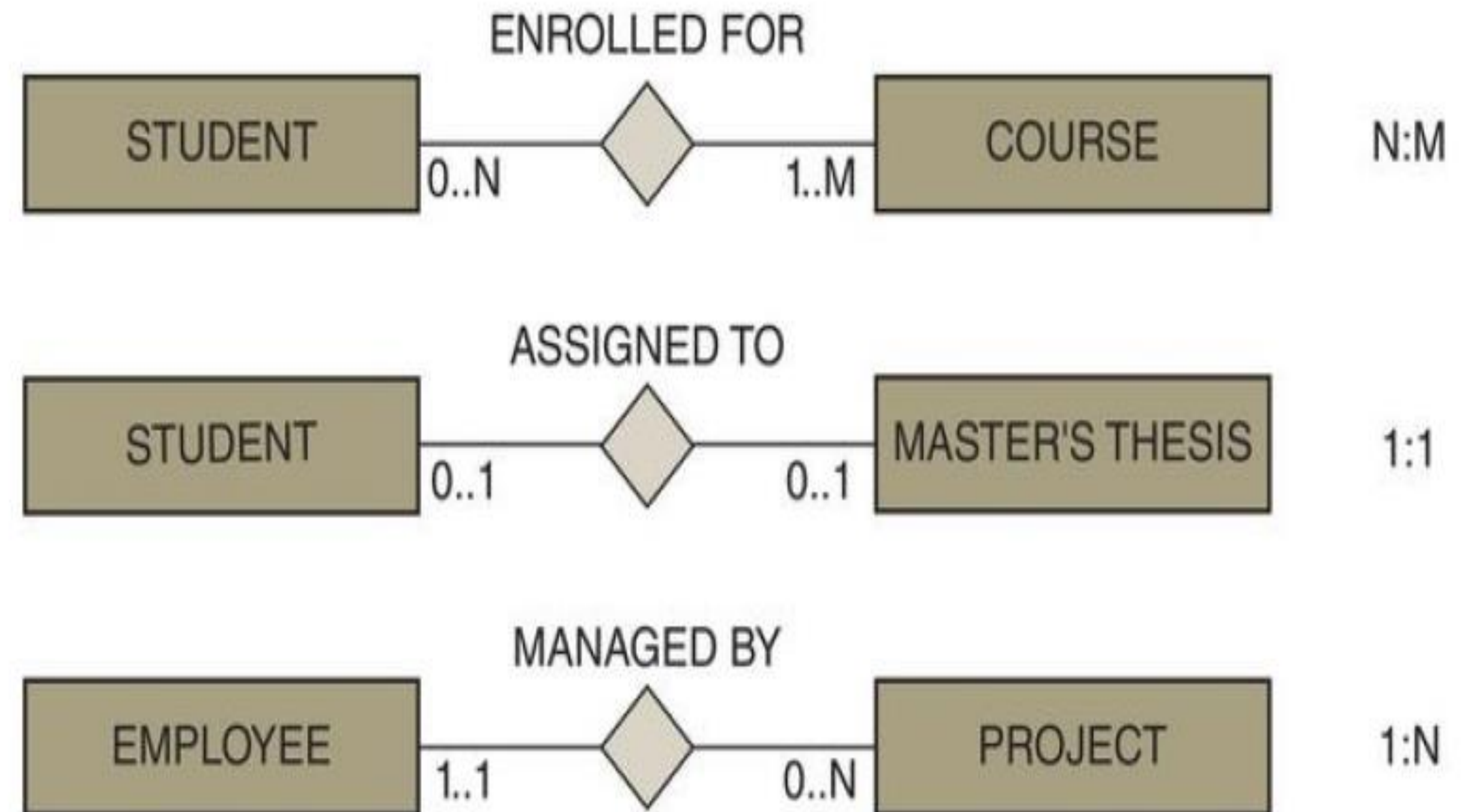


# The entity relationship model



An **ENTITY TYPE** represents a business concept with an unambiguous meaning to a particular set of users.  
Examples?

Every relationship type can be characterized in terms of its cardinalities, which specify the minimum or maximum number of relationship instances that an individual entity can participate in.



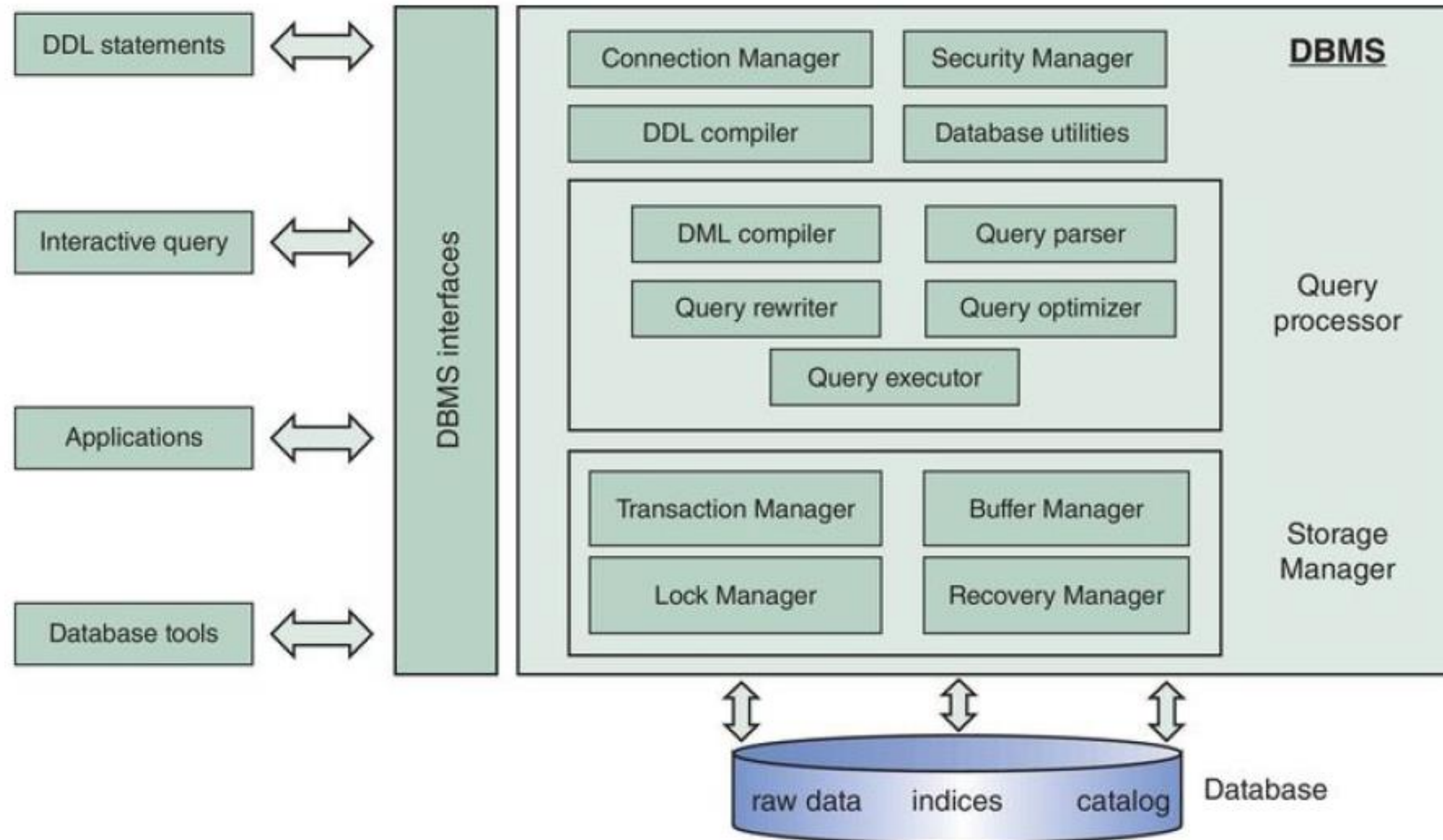
Example: A student is **enrolled for** 1 or M courses

- DBMS Architecture
- Entity Relationship Diagram (ERD)
- Crow's foot notation

# DBMS Archicecture



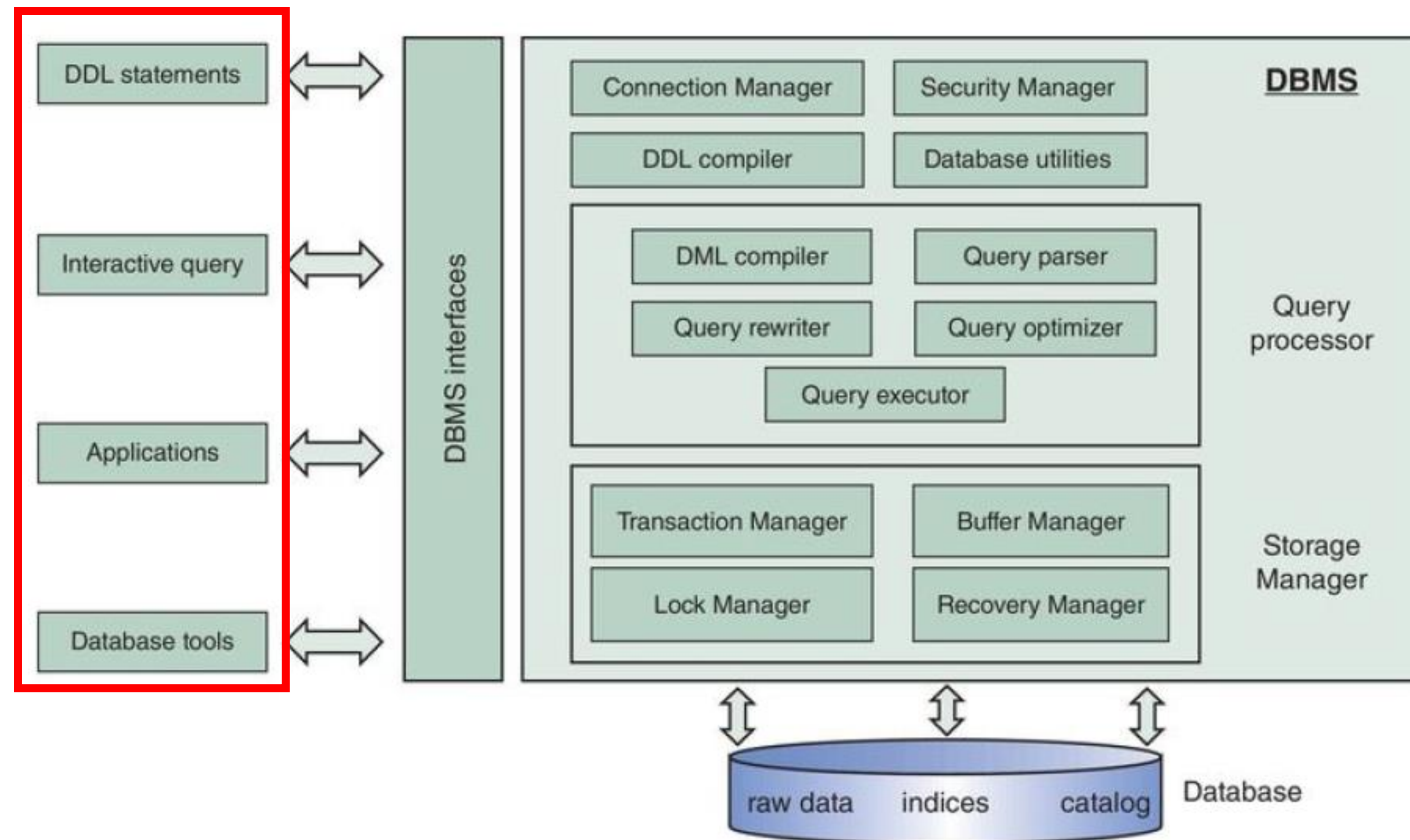
# Architecture of a DBMS





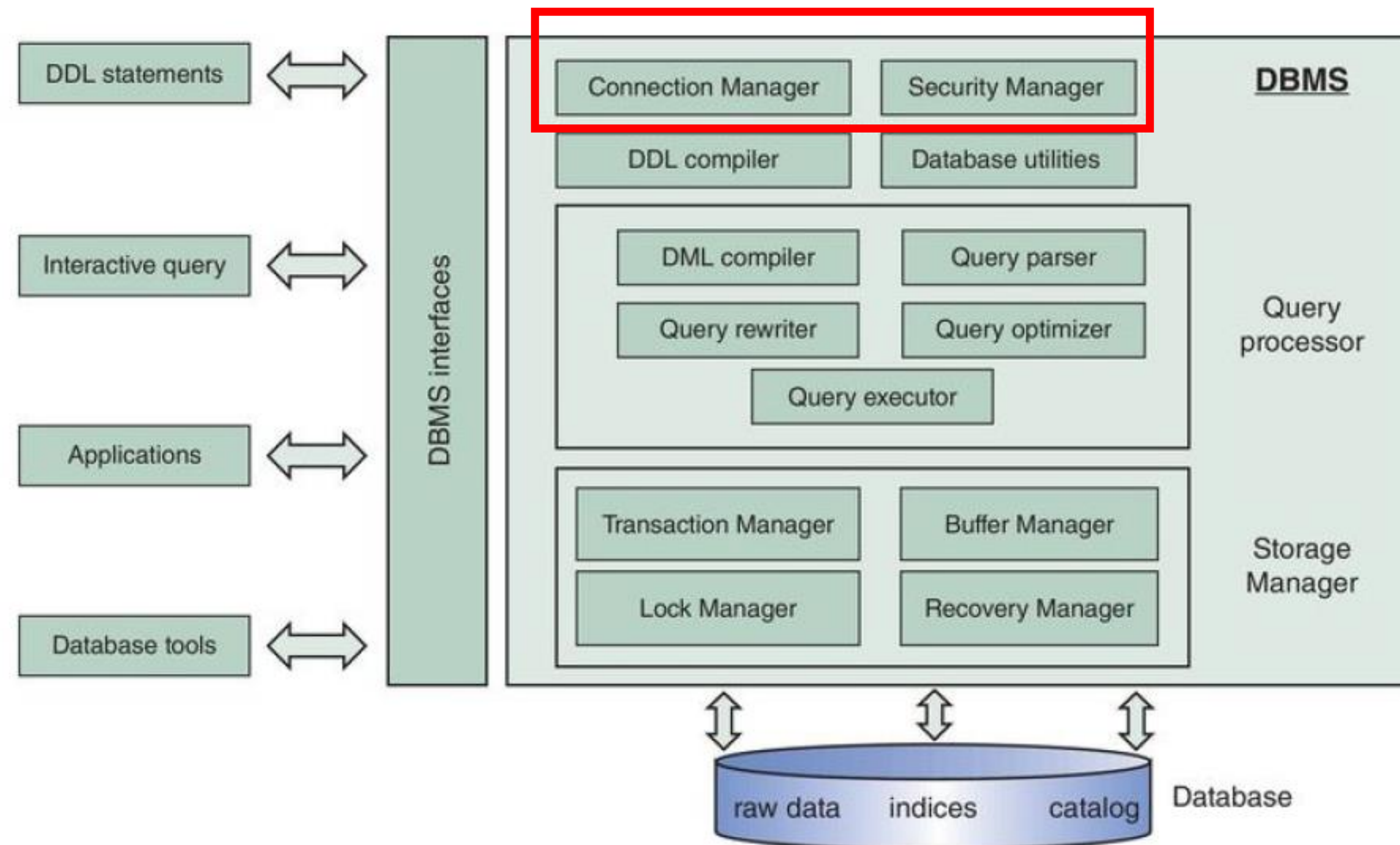
# Architecture of a DBMS

Various ways of interacting with the DBMS. Command line interface, form-base interface, tools to maintain or fine-tune the DBMS.



# Architecture of a DBMS

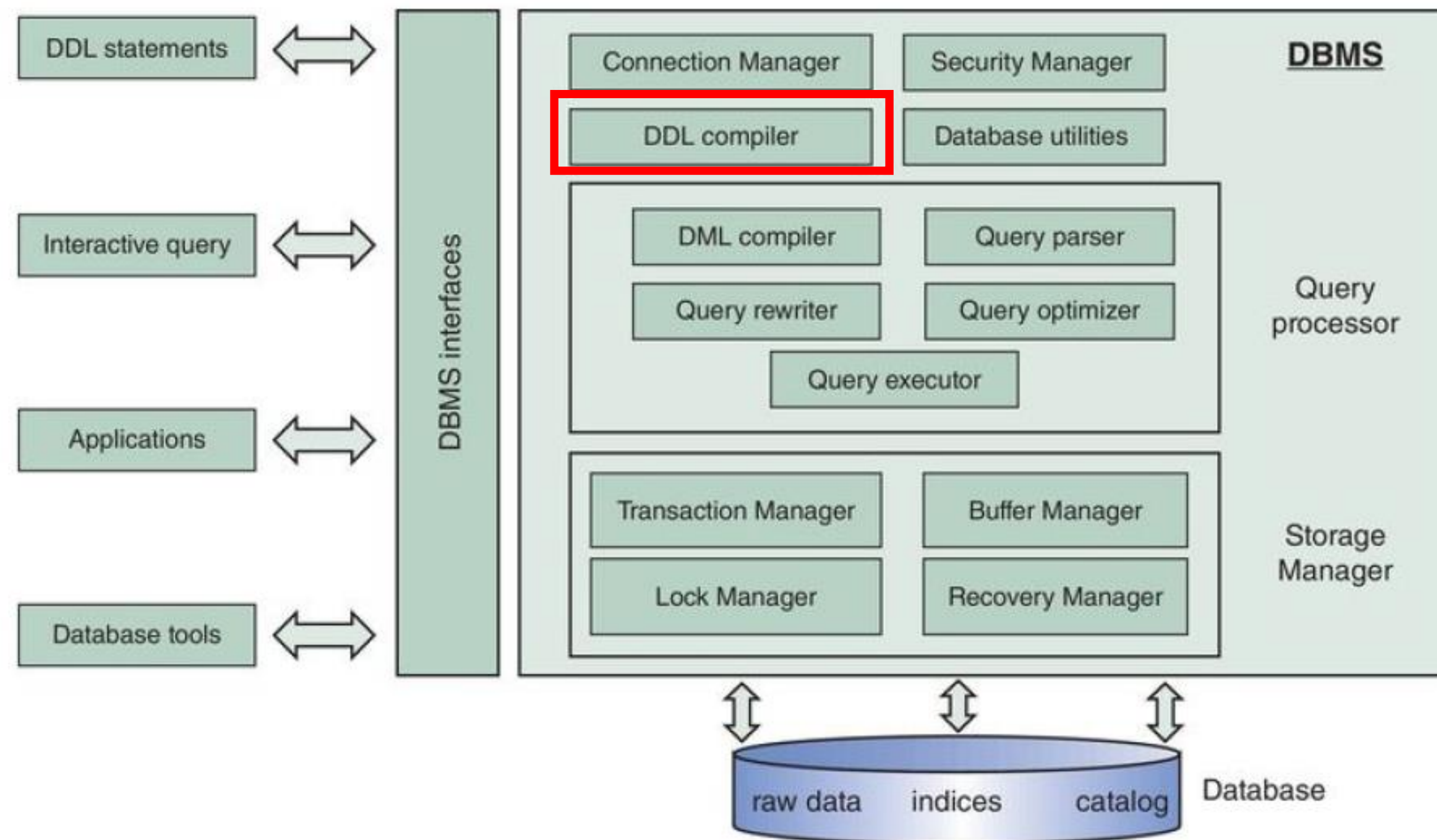
The **connection manager** provides facilities to set-up a database connection. It can be set-up locally or through a network. The **security manager** verifies whether a user has the right privileges to execute the database actions required.





# Architecture of a DBMS

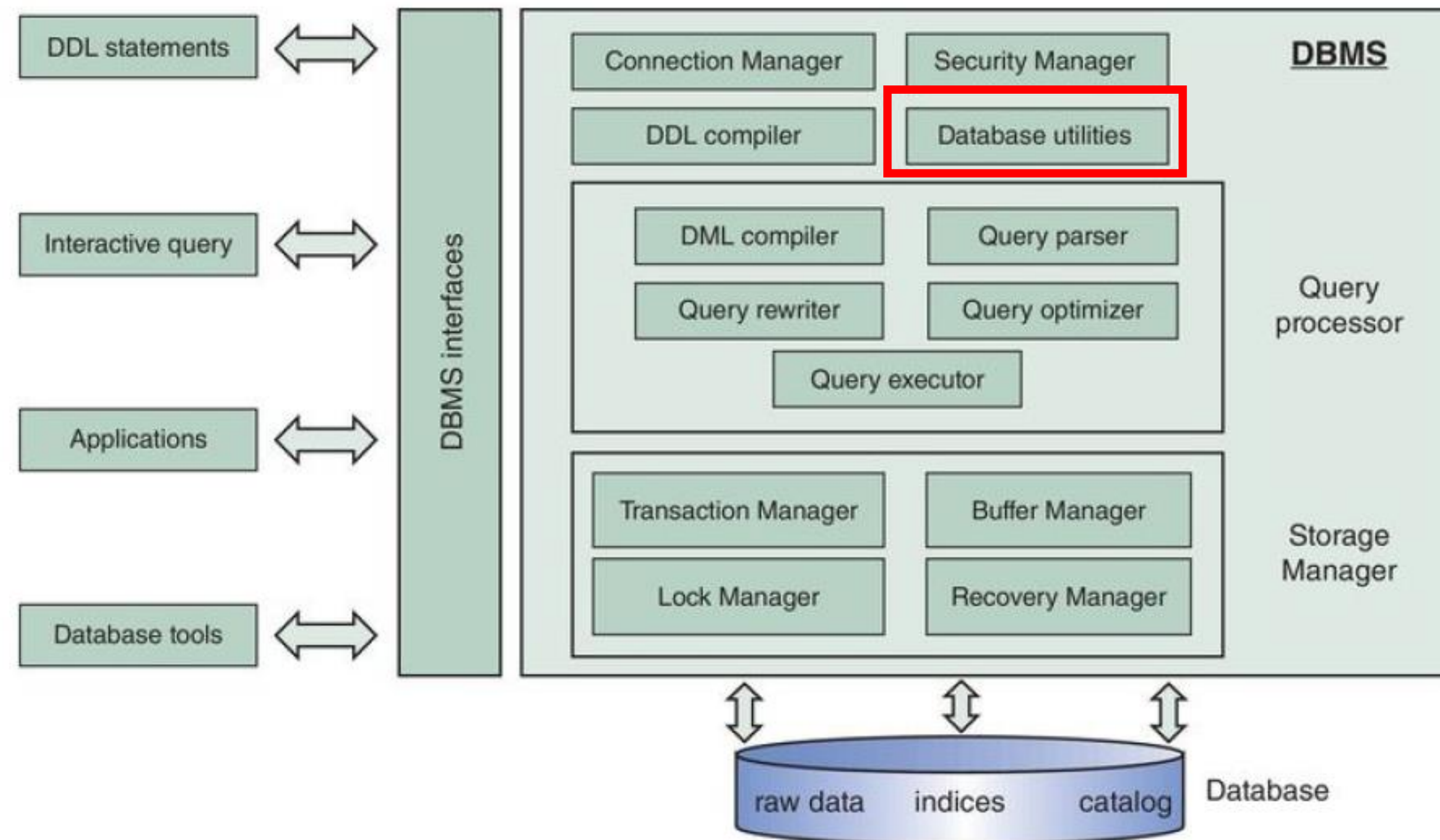
The **Data Definition Language (DDL) compiler** compiles the data definitions specified in DDL. Most relational databases use SQL as their DDL.





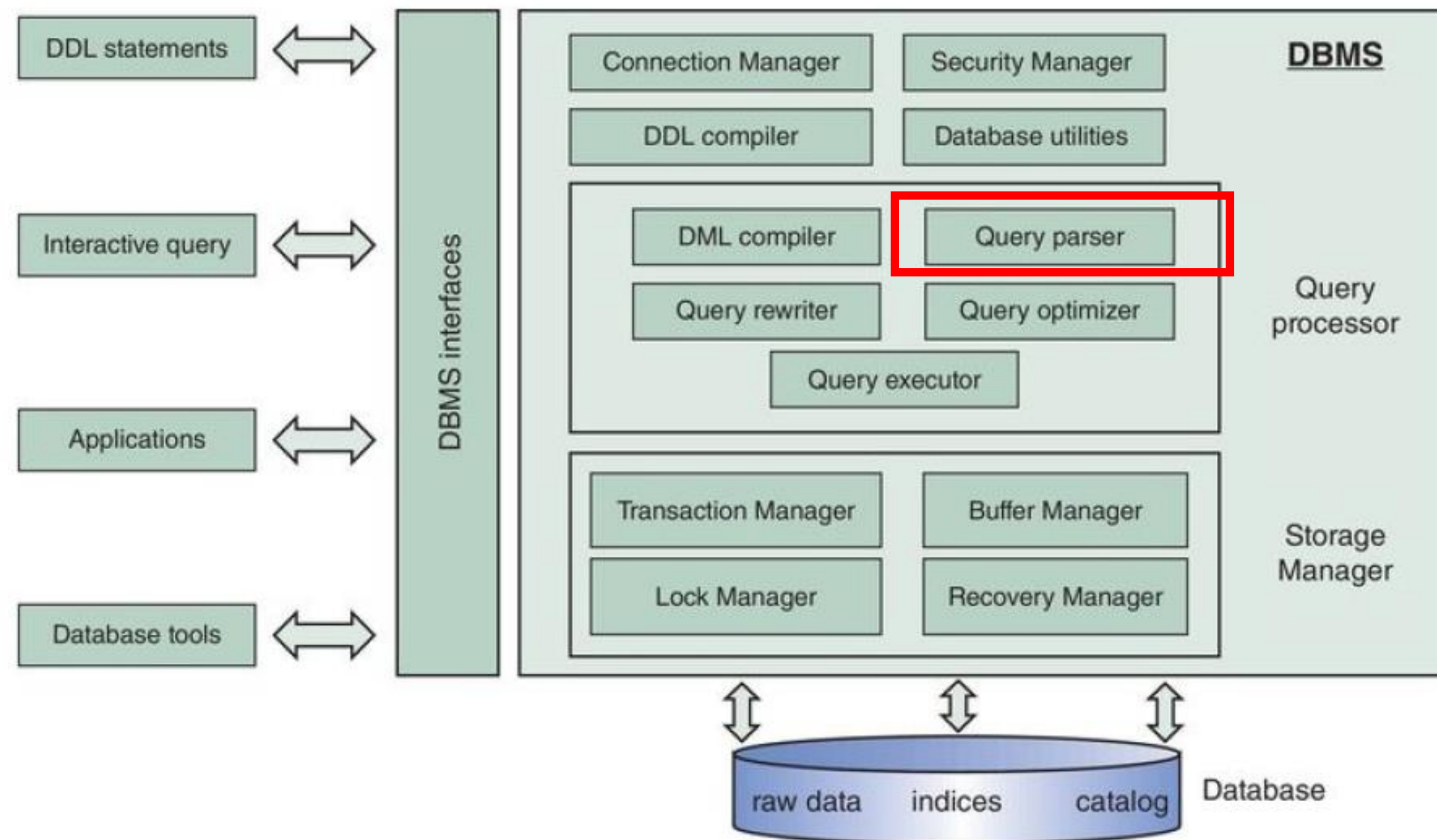
# Architecture of a DBMS

Various utilities, example: A loading utility (load data from a variety of sources), reorganization utility (reorganizes the data), user management utilities (support the creation of user groups or accounts), etc.



# Architecture of a DBMS

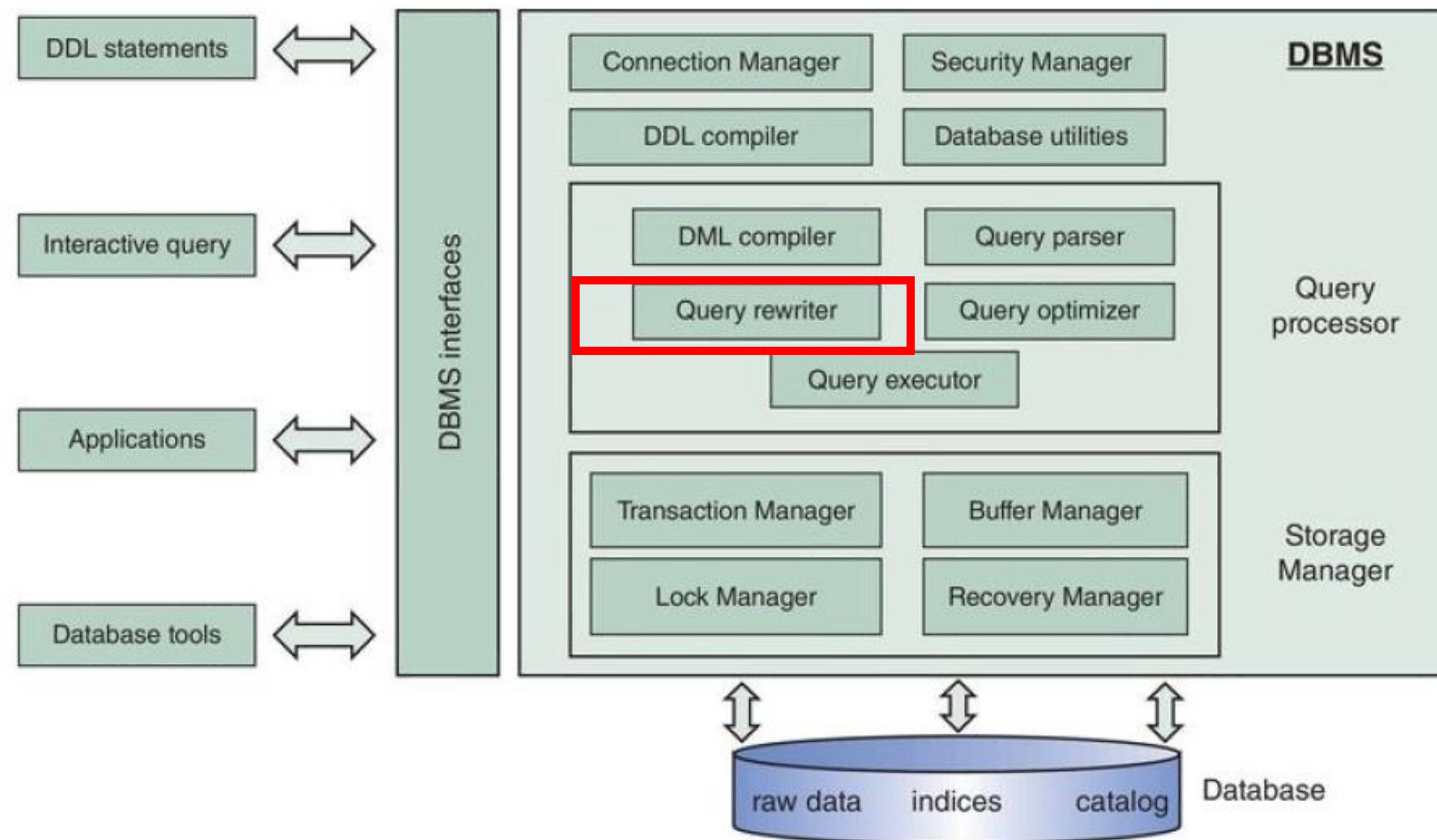
The **query parser** parses the query into an *internal representation format* that can then be further evaluated by the system. It checks the query for syntactical and semantical correctness.





# Architecture of a DBMS

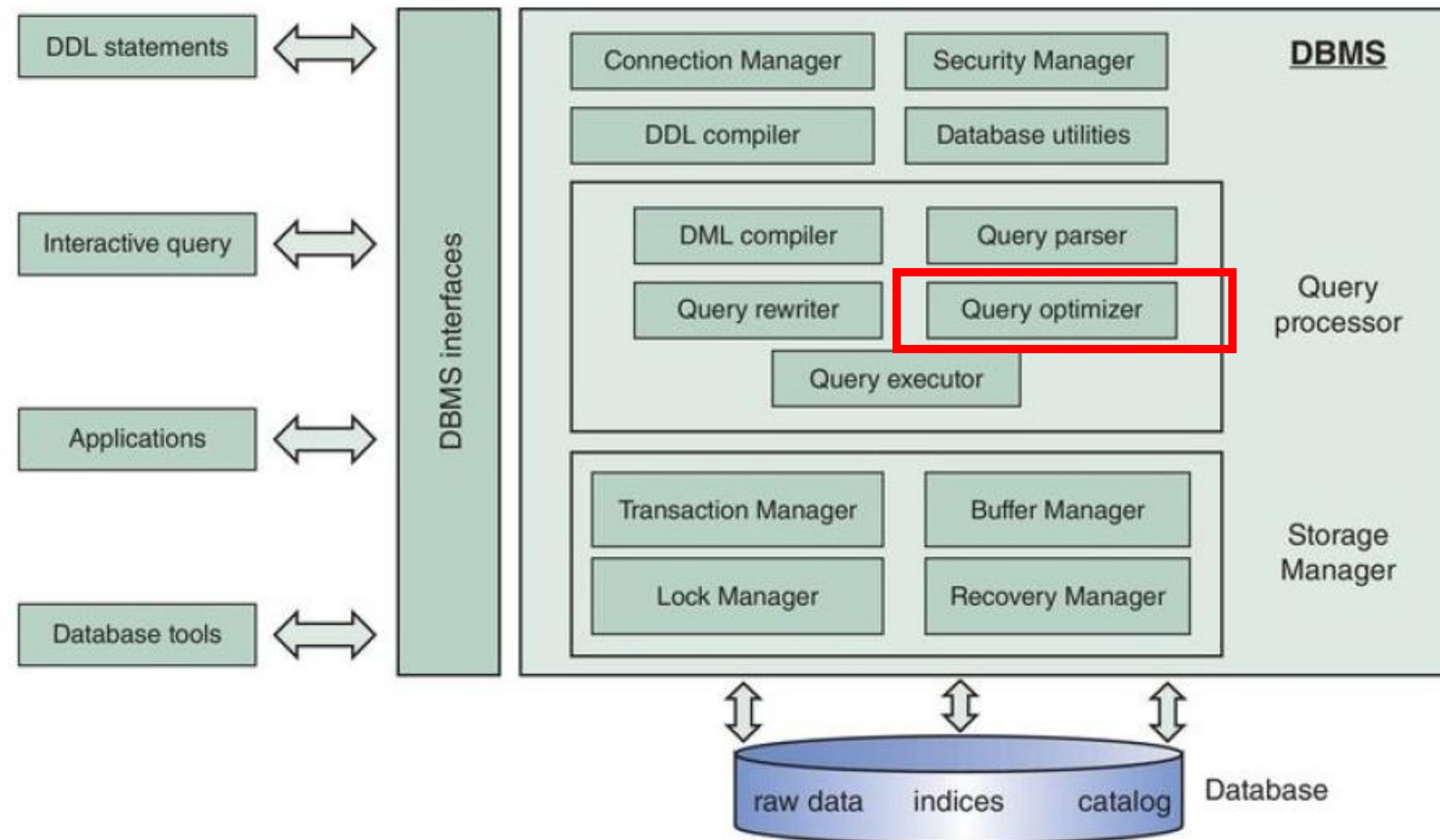
The **query rewriter** optimizes the query, independently of the current database state. It simplifies it using a set of predefined rules and heuristics that are DBMS-specific.





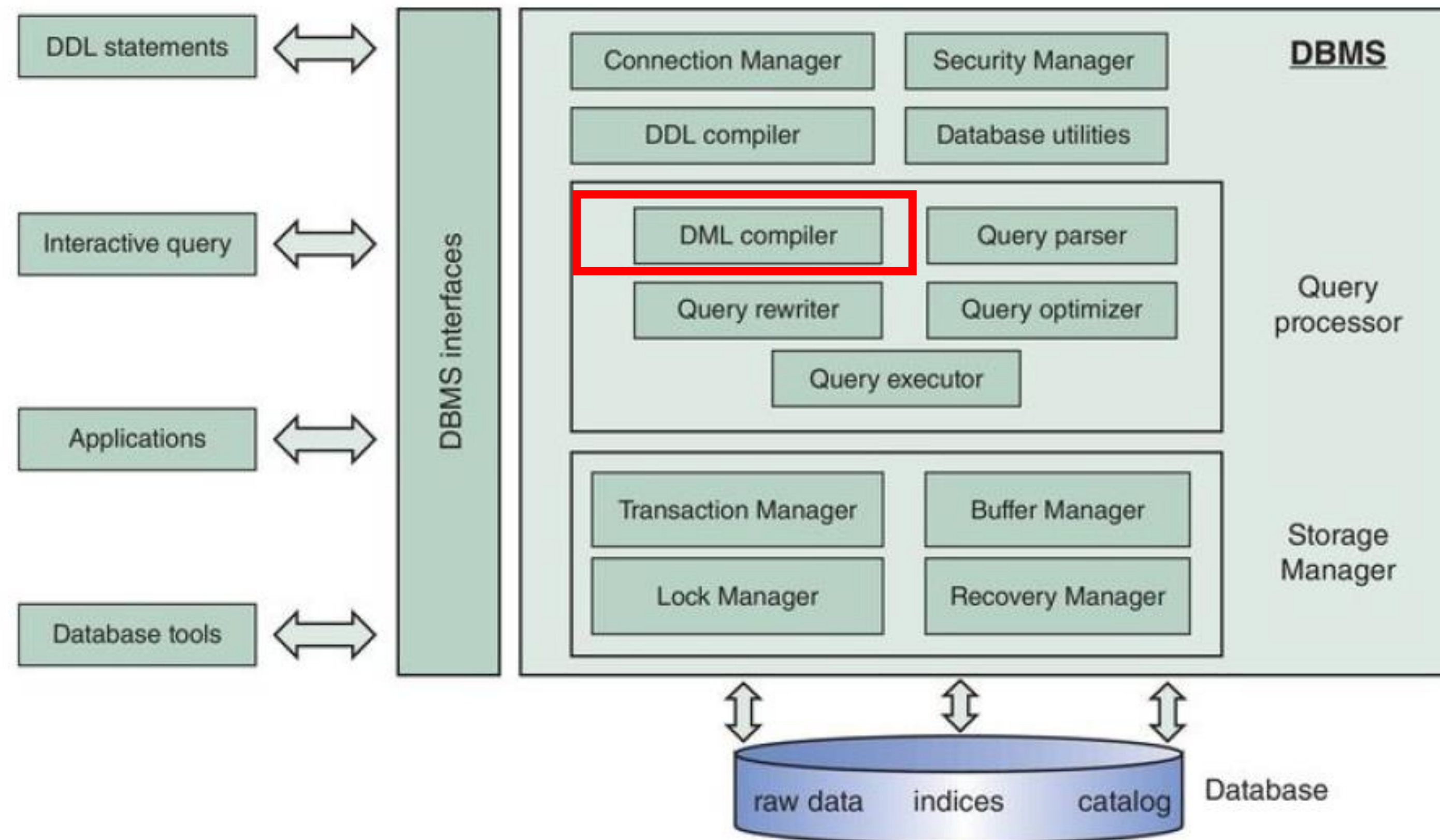
# Architecture of a DBMS

The **query optimizer** optimizes the query based upon the current database state. It can make use of predefined indexes that are part of the internal data model and provide quick access to the data.



# Architecture of a DBMS

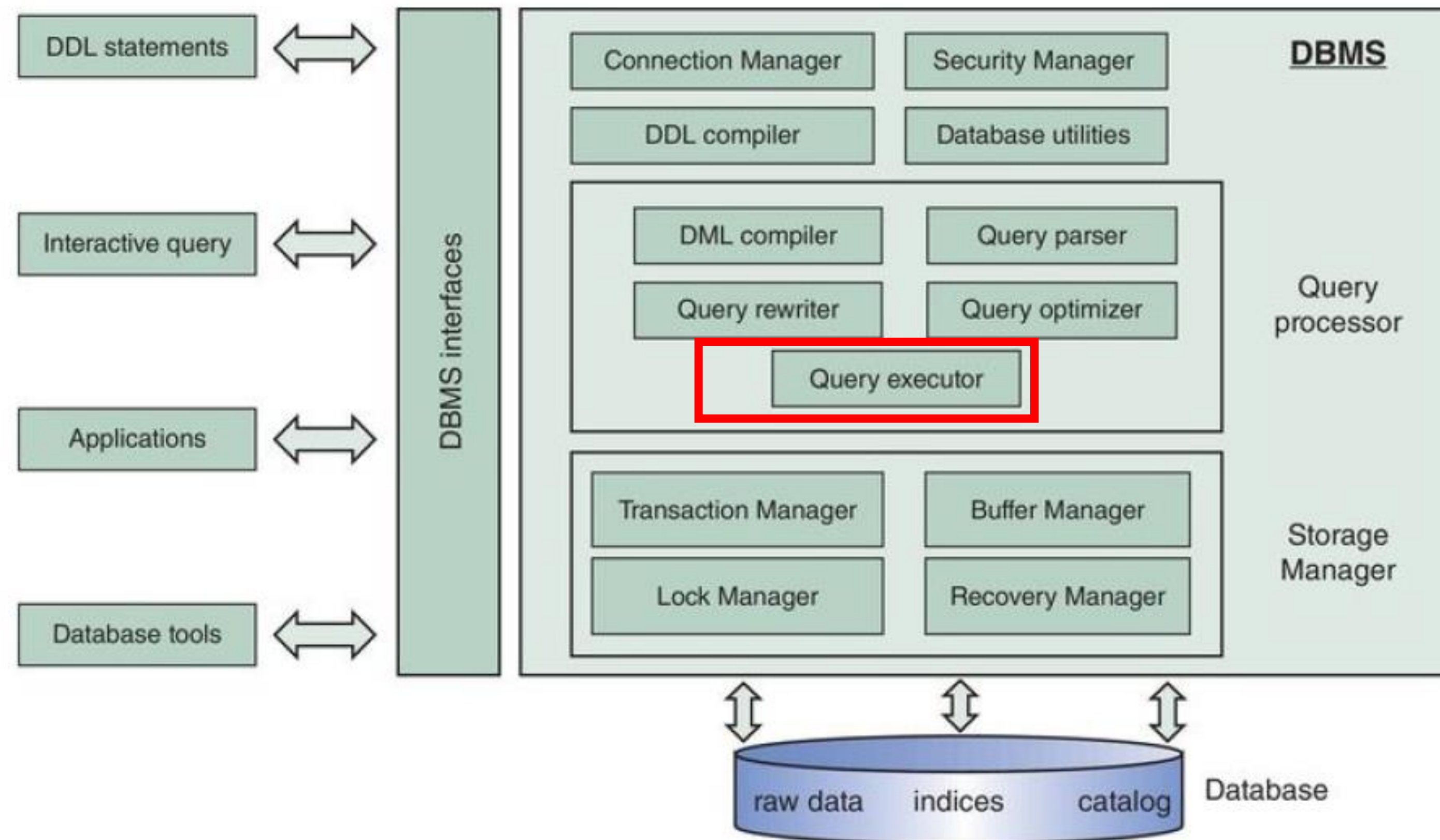
The **Data Manipulation Language (DML) compiler** compiles the data manipulation statements specified in DML.





# Architecture of a DBMS

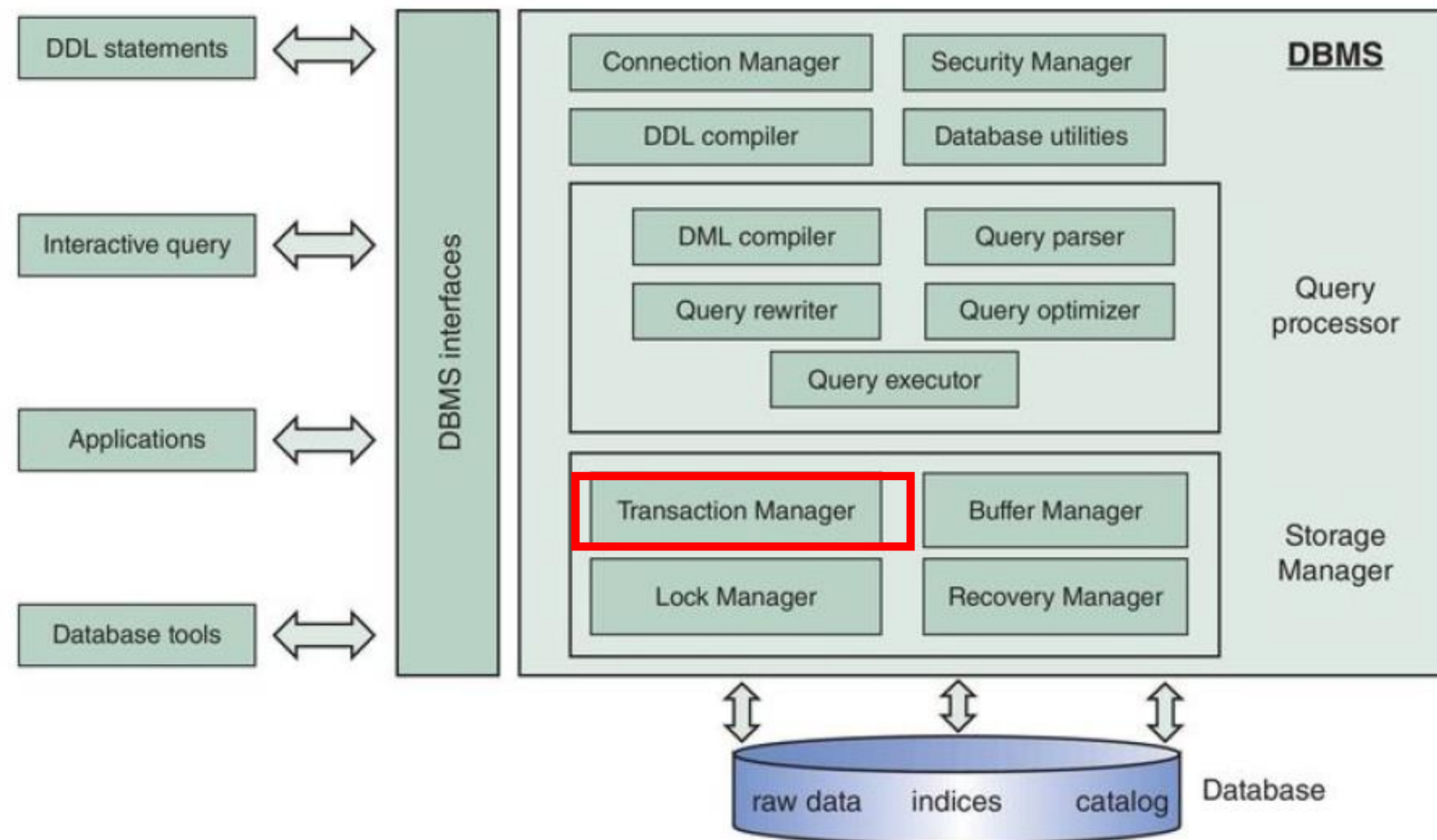
The **query executor** takes care of the actual execution by calling on the storage manager to retrieve the data requested





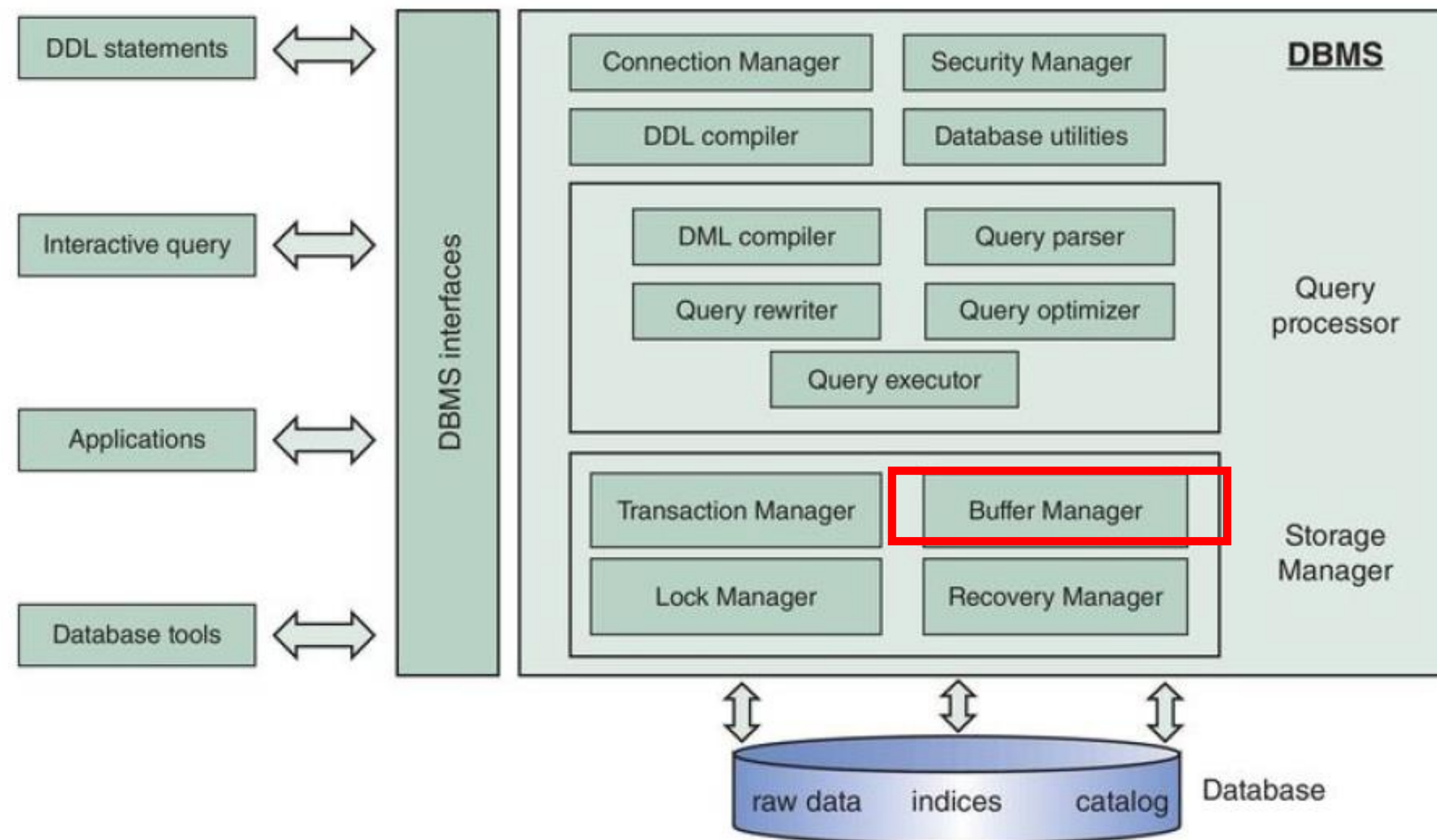
# Architecture of a DBMS

The **transaction manager** supervises the execution of database transactions. Remember, a database transaction is a sequence of read/write operations considered to be an atomic unit.



# Architecture of a DBMS

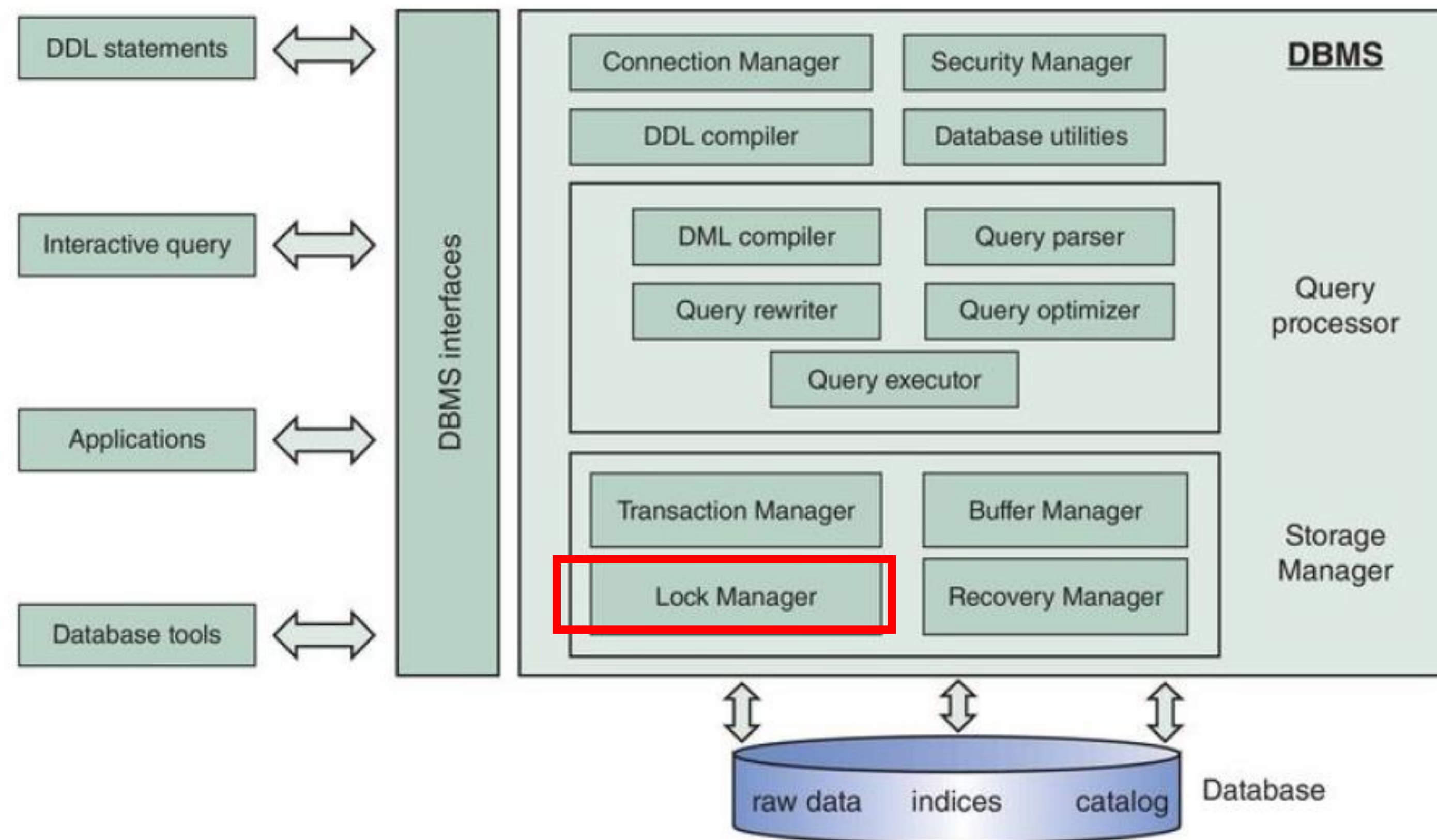
The **buffer manager** is responsible for managing the buffer memory of the DBMS. The DBMS checks first the memory when data need to be retrieved. Retrieving data from the buffer is significantly faster than retrieving them from external disk-based storage.





# Architecture of a DBMS

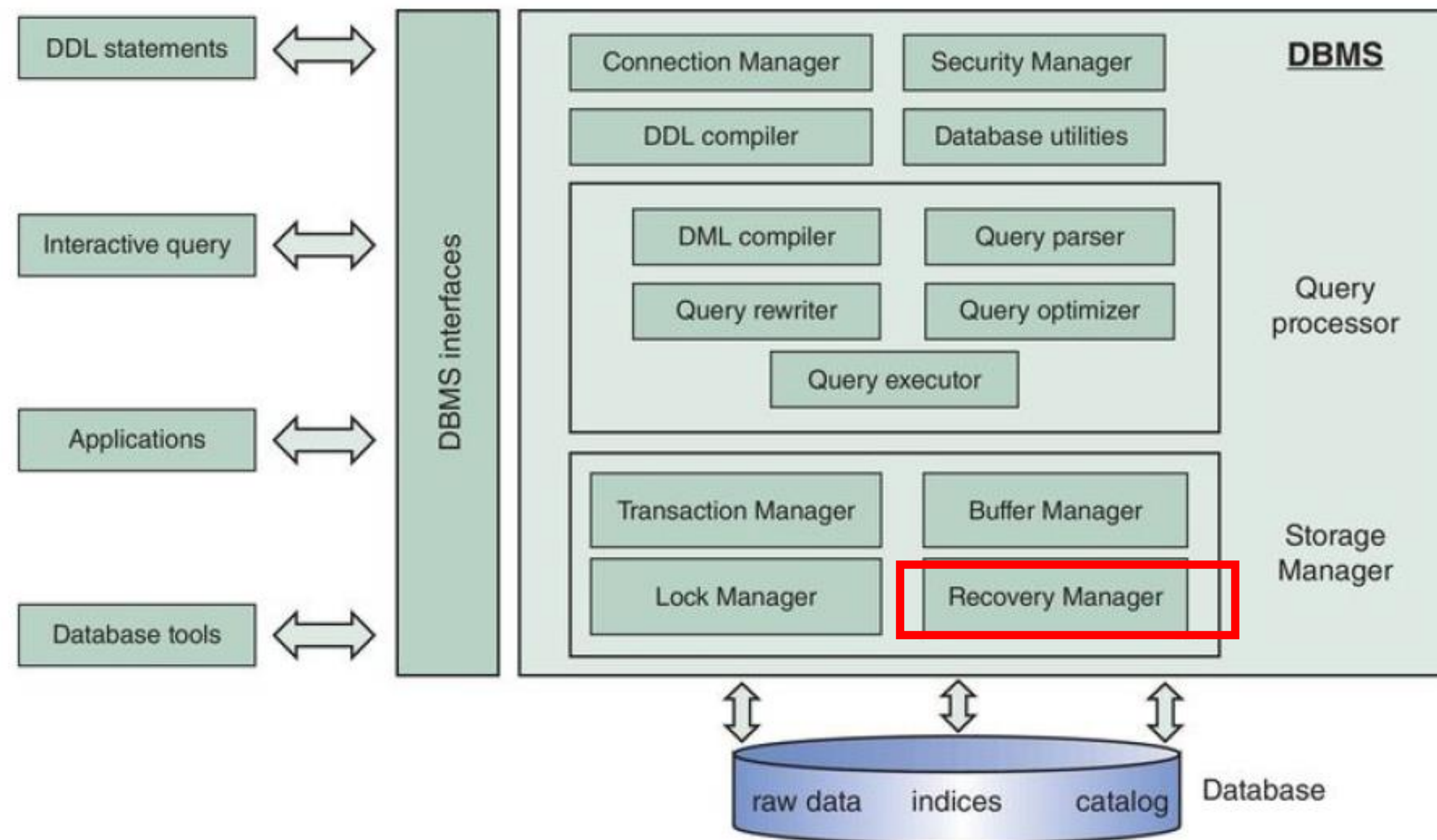
The **lock manager** is an essential component for providing concurrency control, which ensures data integrity at all times.





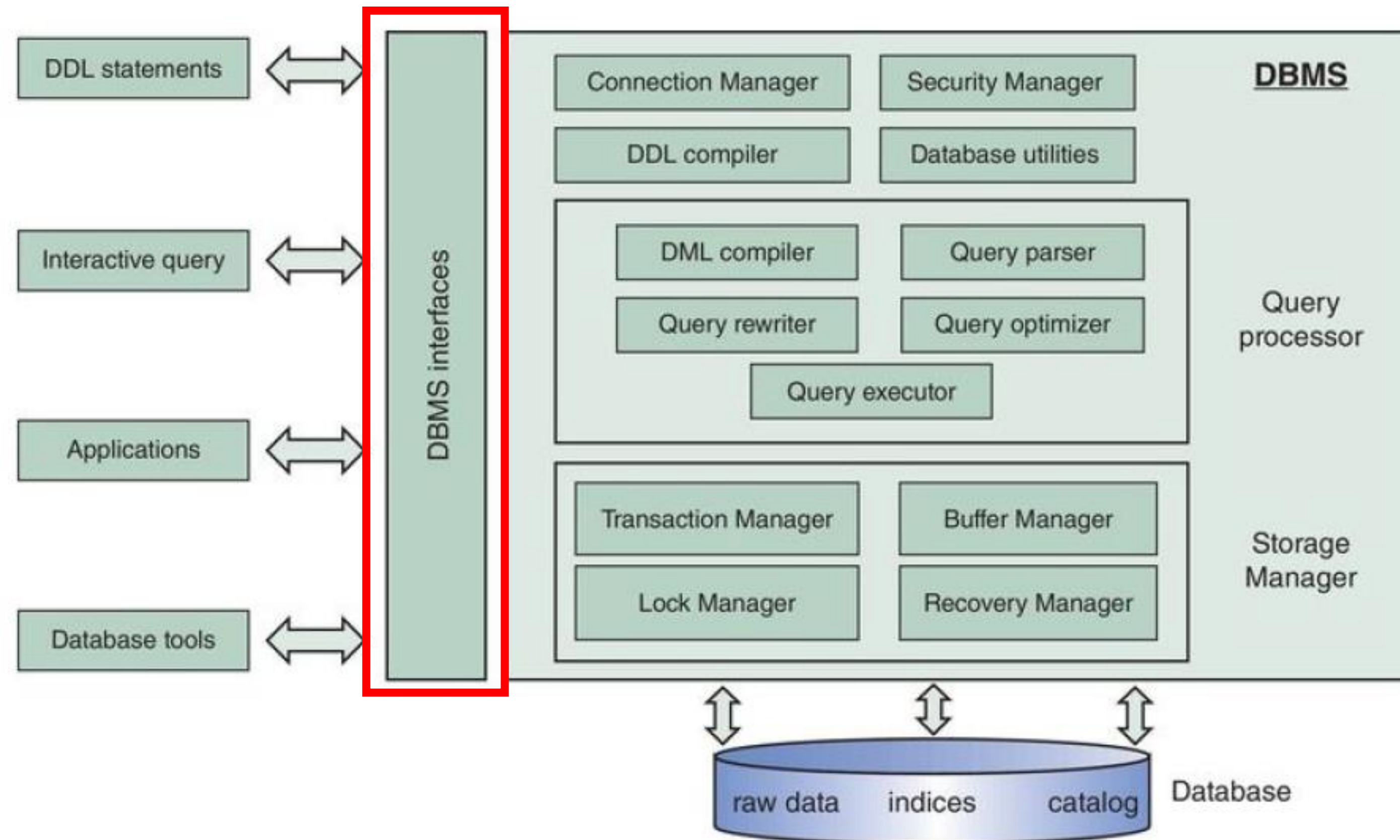
# Architecture of a DBMS

The **recovery manager** supervises the correct execution of database transactions. It keeps track of all database operations in a **logfile** and will be called upon to undo actions of aborted transactions or during crash recovery.



# Architecture of a DBMS

A DBMS needs to interact with various parties, such as a database designer, a database administrator, an application, or even an end-user (Ex. APIs)



## Review question

Fill in the gaps in the following sentences:

When, during crash recovery, aborted transactions need to be undone, that is a task of the ...**A**...

The part of the storage manager that guarantees the ACID properties is the ...**B**...

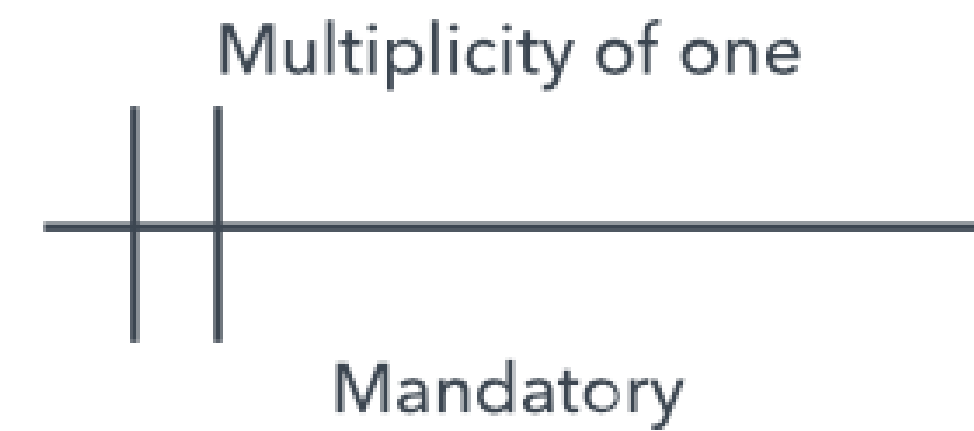
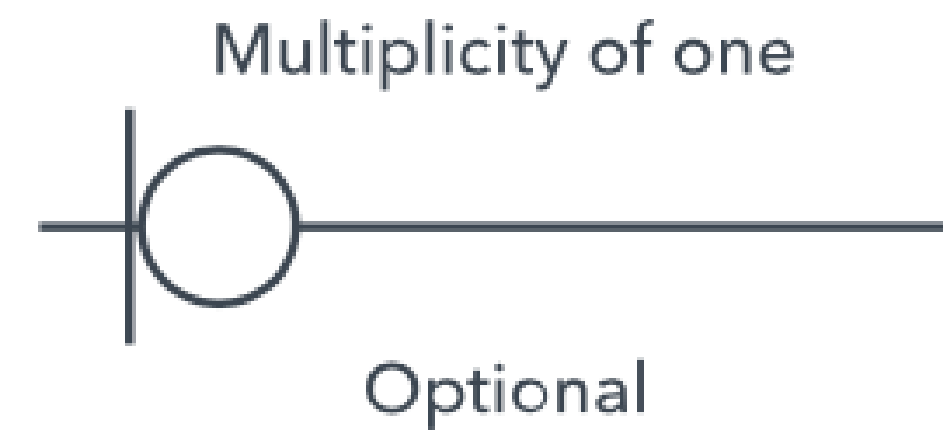
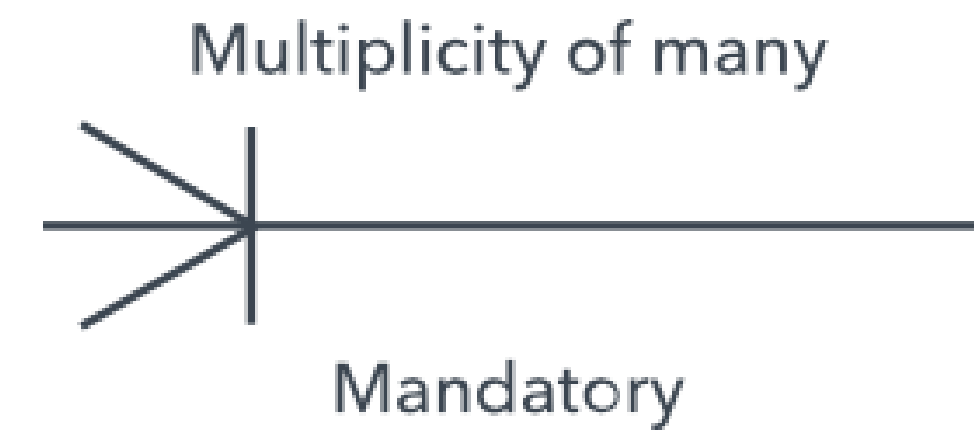
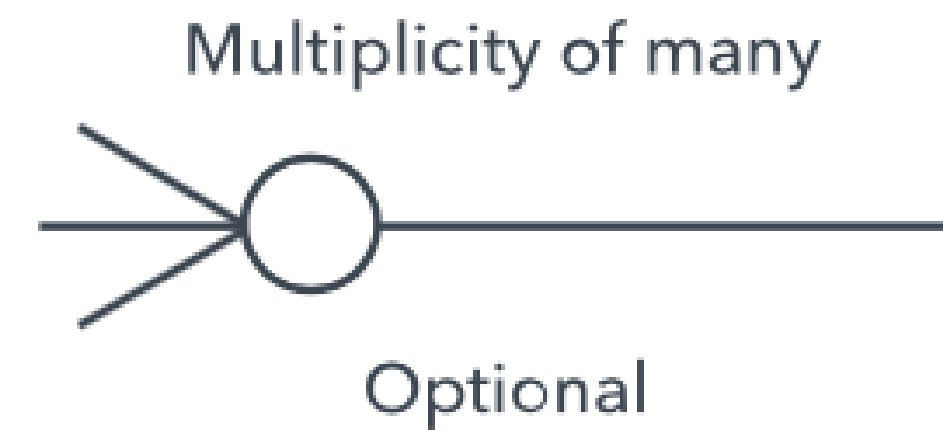
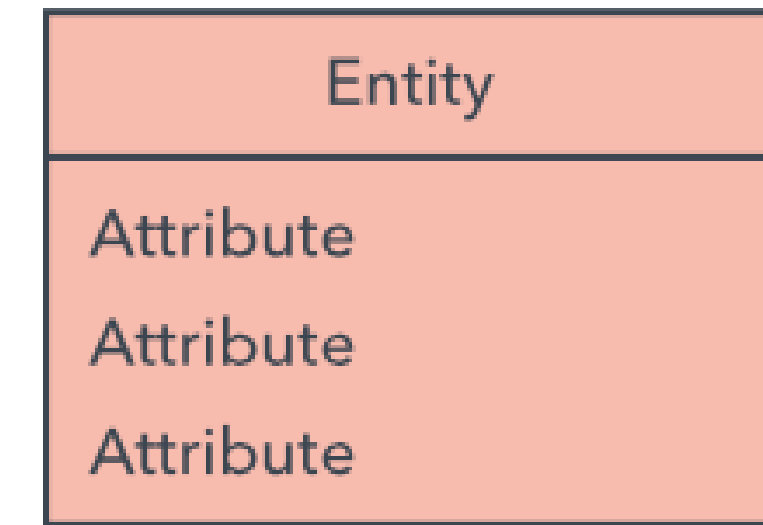
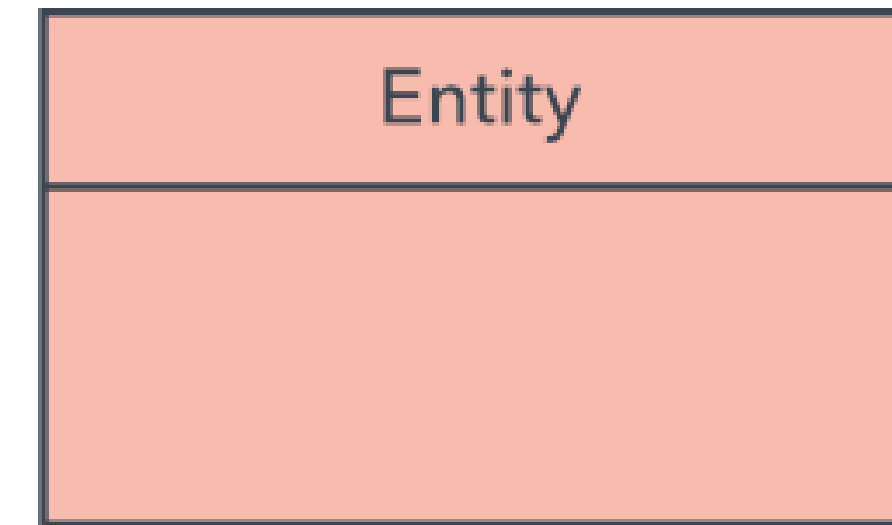
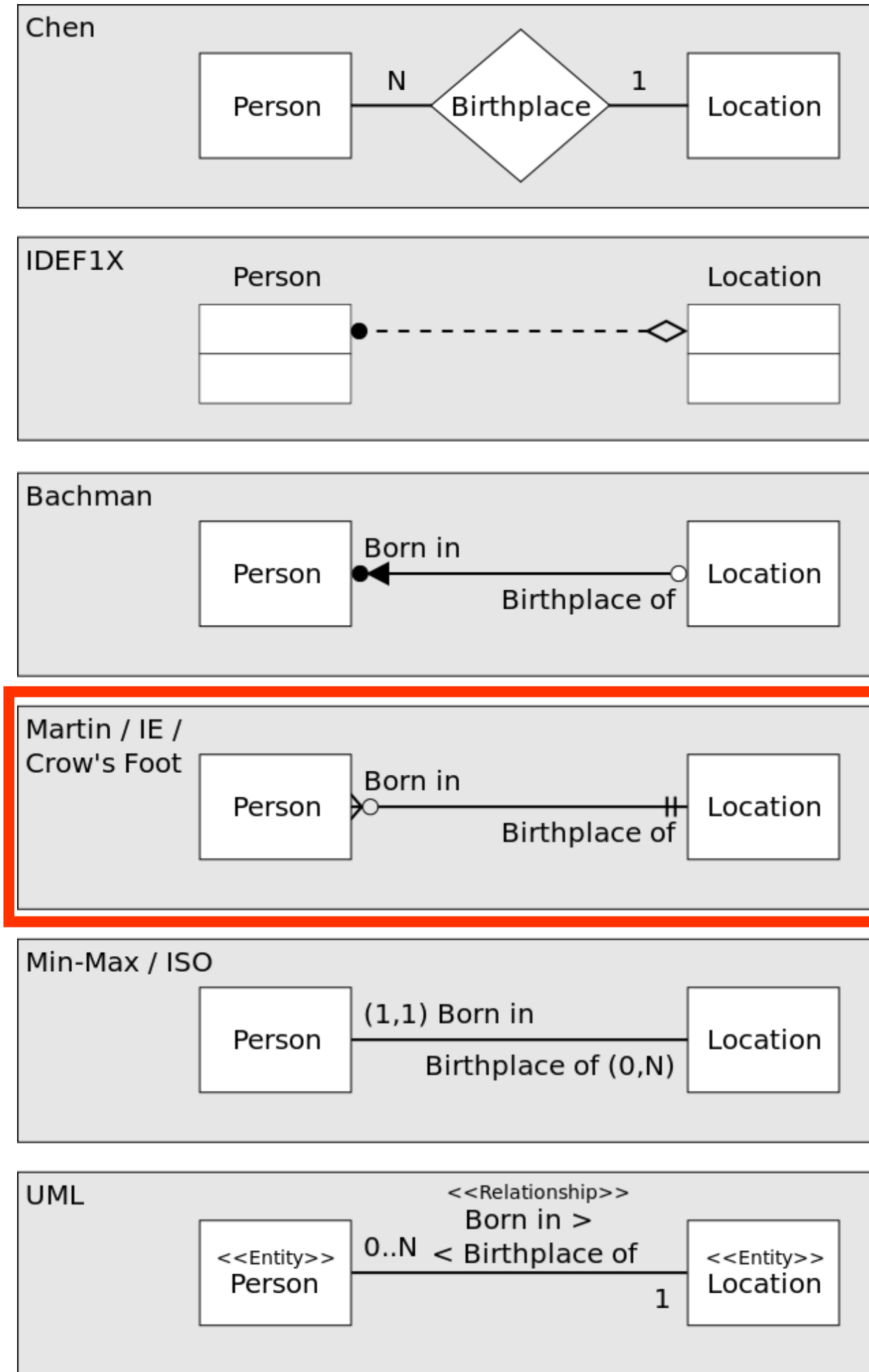
- a. A: lock manager, B: recovery manager.
- b. A: lock manager, B: lock manager.
- c. A: recovery manager, B: buffer manager.
- d. A: recovery manager, B: transaction manager.



# Relational Modelling

## Entity Relationship Diagram (ERD)

# Different types of Notations for the ERD



# CROW's foot notation



Gordon Everest proposed the Crow's Foot notation in a 1976 paper. Several development methodologies enhanced the notation in the 1980s. Most Computer-Aided Software Engineering (CASE) tools support some variation of the Crow's Foot notation.

Source: Mannino, M. V. (2019). *Database design, application development, and administration*. 7<sup>th</sup> edition. Chicago Business Press



# CROW's foot notation - Illustrating Basic Symbols

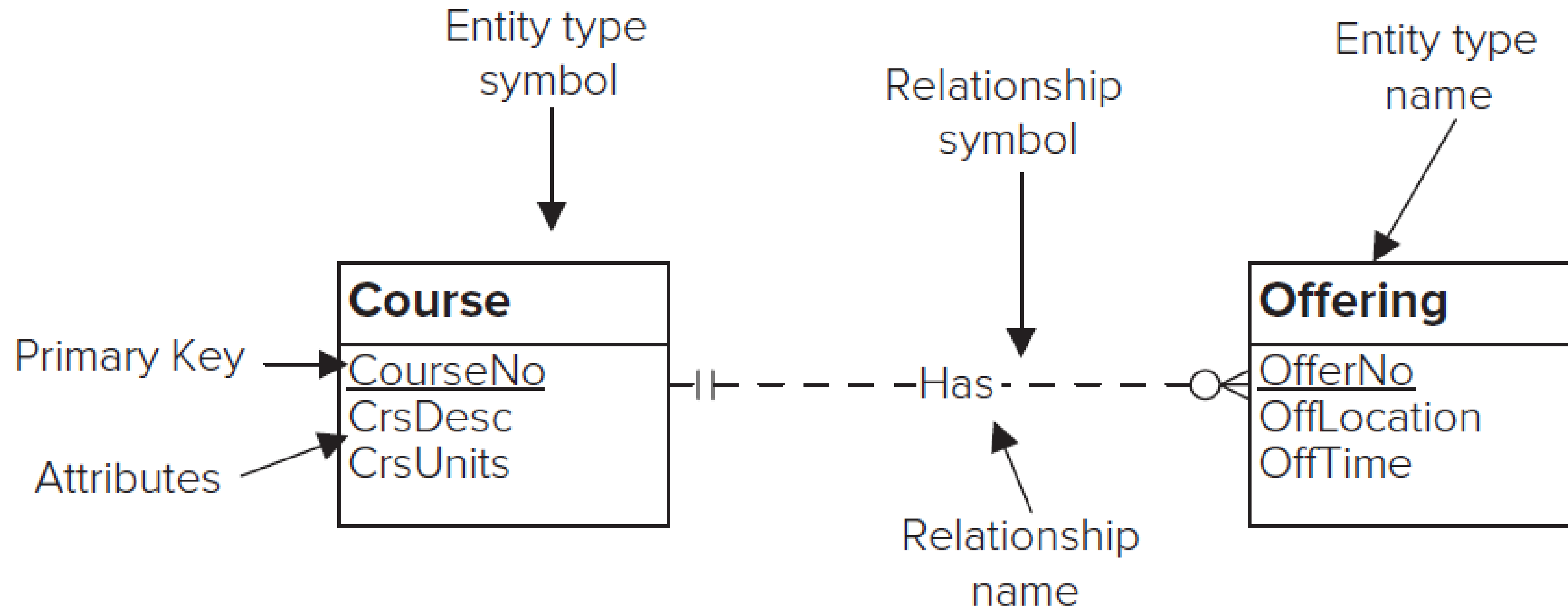
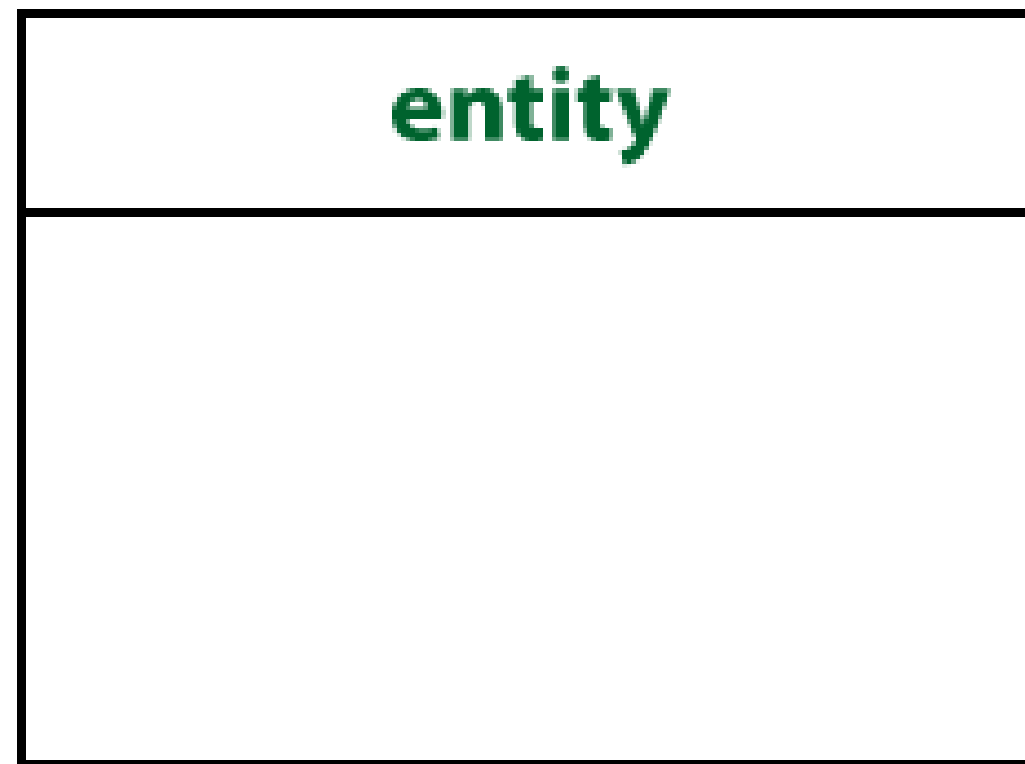
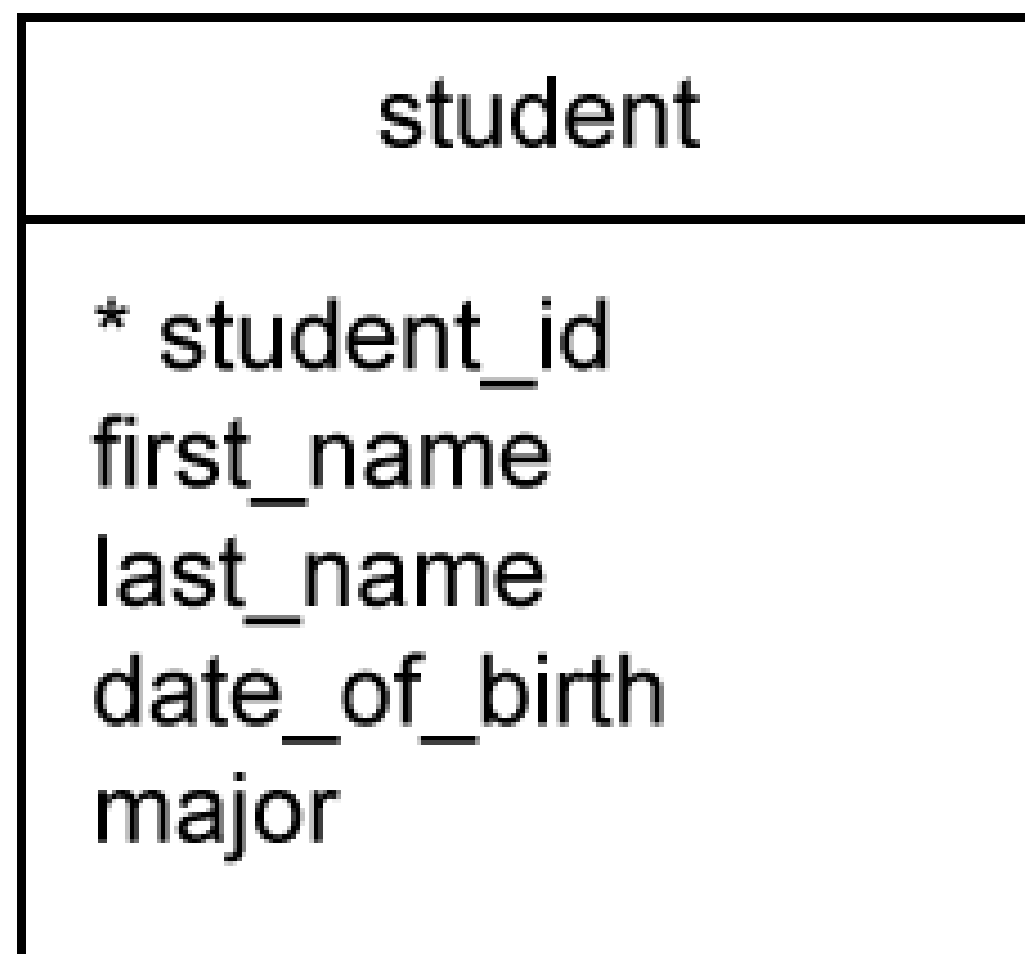


Image Source: Mannino, M. V. (2005). *Database design, application development, and administration*. 7<sup>th</sup> edition. Chicago Business Press

# CROW's foot notation - Notation



An entity is represented by a rectangle, with its name on the top. The name is singular.



} **attributes**

The attribute(s) that uniquely distinguishes an instance of the entity is the identifier.

Image Source: <https://www.vertabelo.com/blog/crow-s-foot-notation/>



# CROW's foot notation - Notation

Relationships have two indicators.

The **first one** refers to the maximum number of times that an instance of one entity can be associated with instances in the related entity.

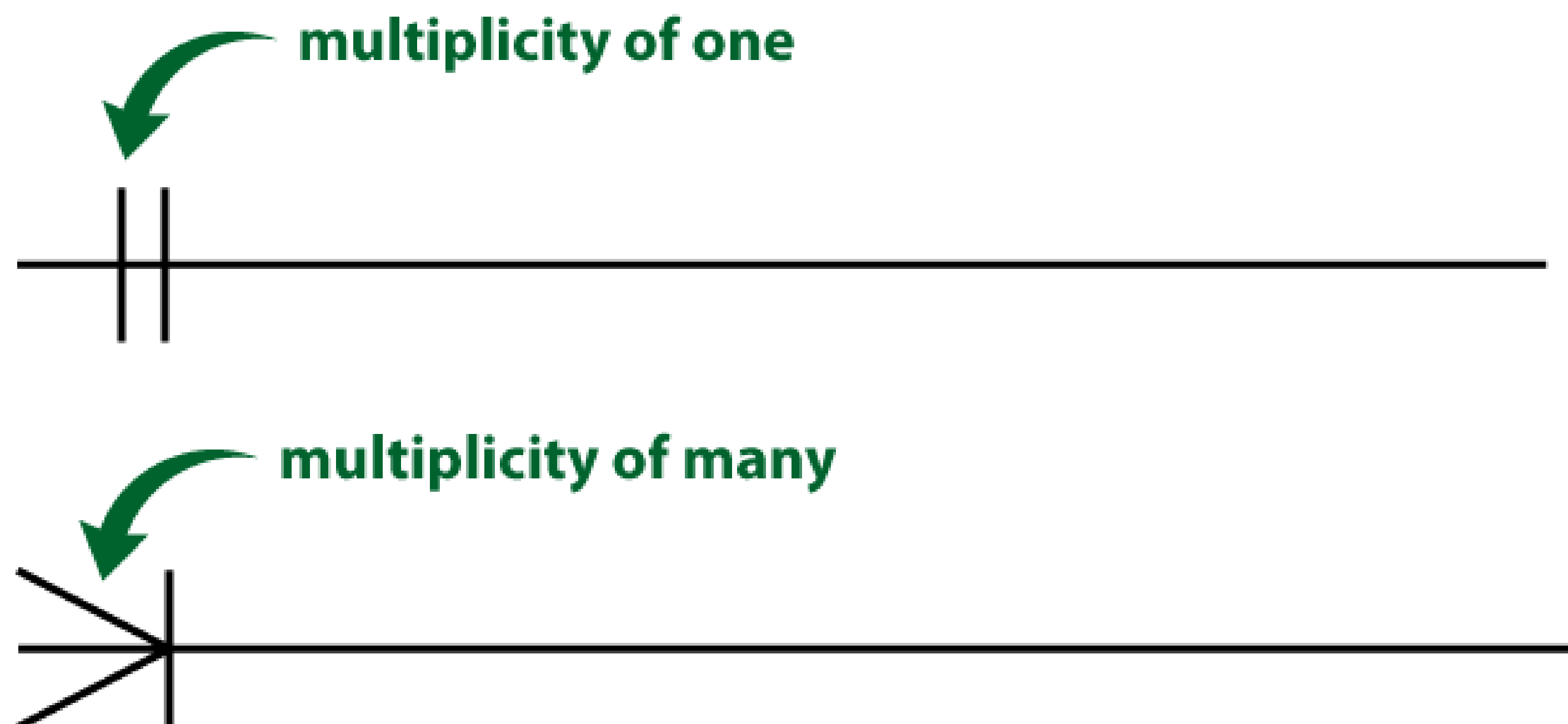


Image Source: <https://www.vertabelo.com/blog/crow-s-foot-notation/>

# CROW's foot notation - Notation

The **second one** describes the minimum number of times one instance can be related to others. It can be zero or one, and optional or mandatory.

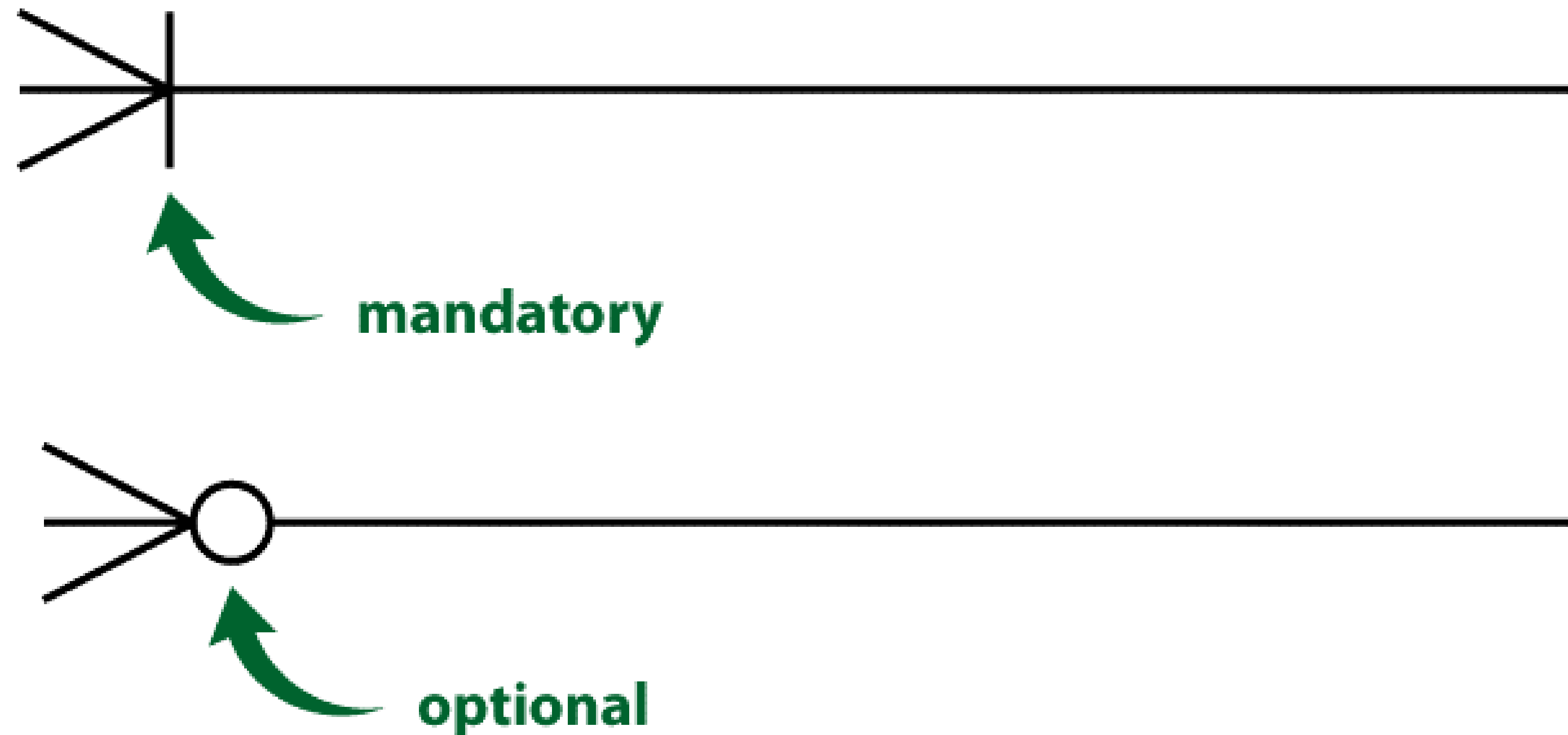
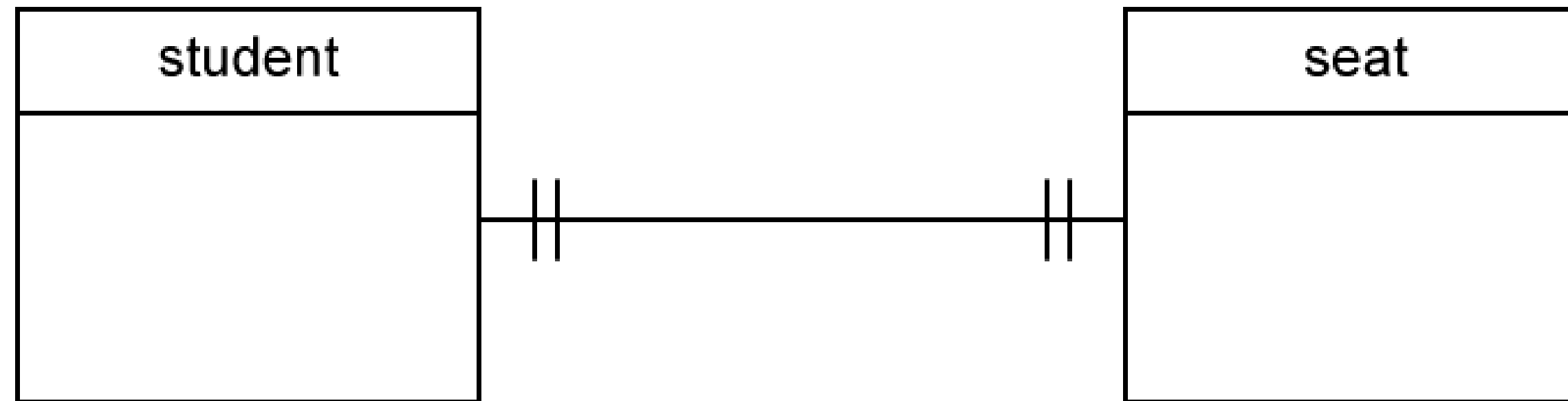


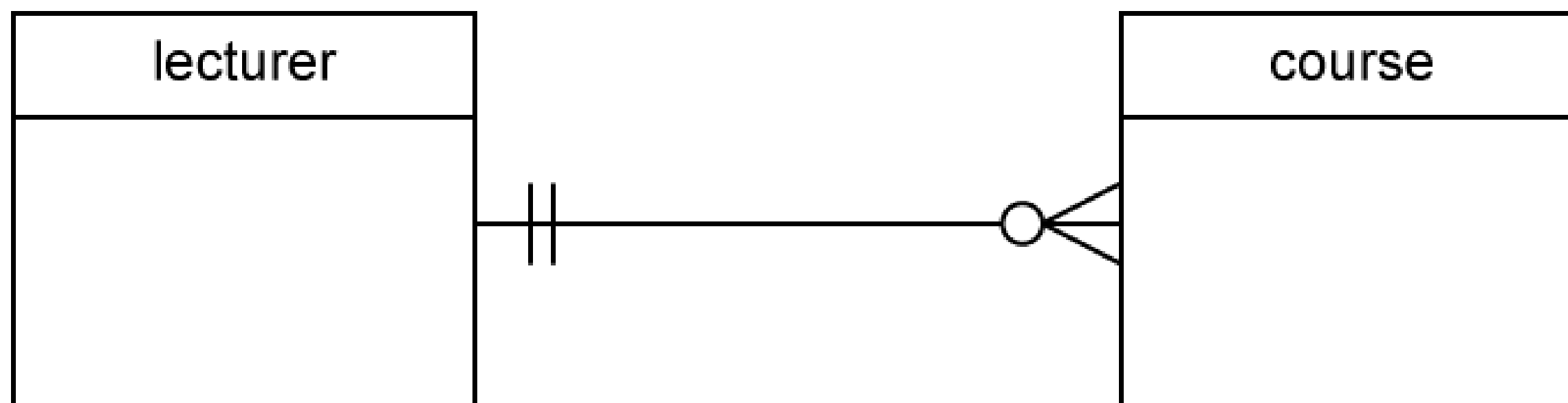
Image Source: <https://www.vertabelo.com/blog/crow-s-foot-notation/>



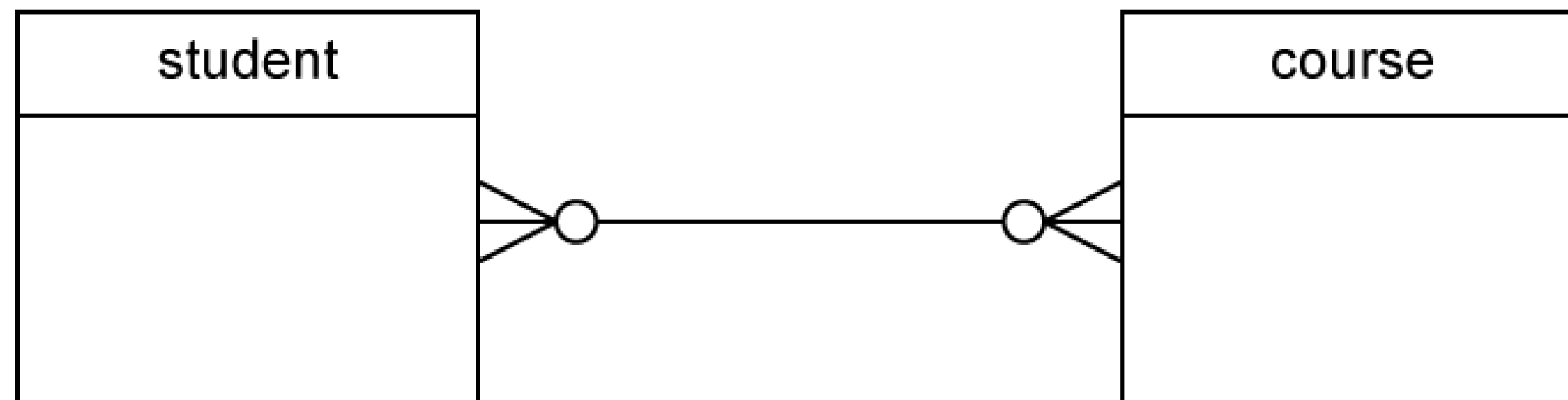
# CROW's foot notation - Examples



One-to-one



One-to-many



Many-to-many

Image Source: <https://www.vertabelo.com/blog/crow-s-foot-notation/>

# CROW's foot notation - Examples

Work in pairs: Consider the following entity types for a patient appointment system:

- Doctor
- Appointment
- Bill
- Payment
- Patient
- Insurance\_company

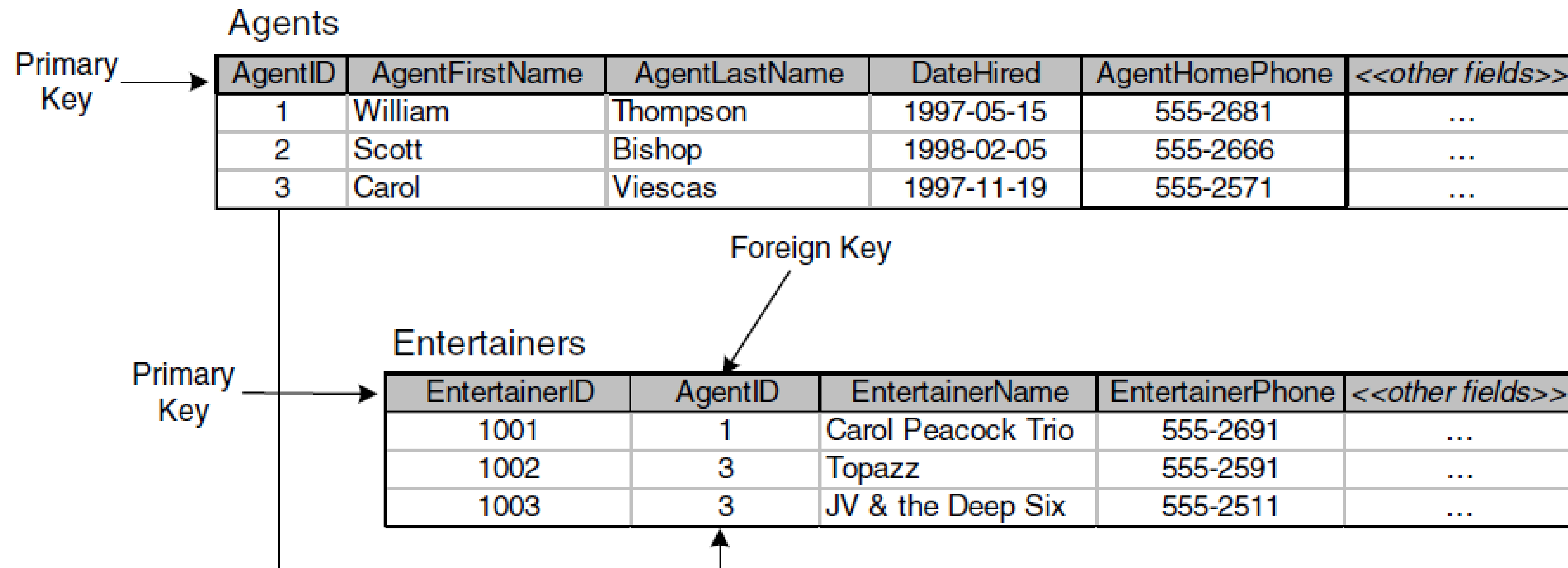
Draw an entity relationship diagram (ERD) with crow's foot notation.  
Several solutions can be valid

<http://www2.cs.uregina.ca/~bernatja/crowsfoot.html>

# From ERD to Tables



## Primary Keys - Unique Identifier of Records in each Table



Foreign Keys - Establish relationships between tables

## Table A



**One to One**

## Table B

## Example:

Agents

AgentID	AgentFirstName	AgentLastName	DateOfHire	AgentHomePhone	<<other fields>>
1	William	Thompson	1997-05-15	555-2681	...
2	Scott	Bishop	1998-02-05	555-2666	...
3	Carol	Viescas	1997-11-19	555-2571	...

Compensation

Salary	CommissionRate	<<other fields>>
\$35,000.00	4.00%	...
\$27,000.00	4.00%	...
\$30,000.00	5.00%	...

to each Agent is linked one and only one Row in Compensation Table



**Table A**


**One to Many**



**Table B**


Example:

Entertainers

EntertainerID	EntertainerName	EntertainerPhone	<<other fields>>
1001	Carol Peacock Trio	555-2691	...
1002	Topazz	555-2591	...
1003	JV & the Deep Six	555-2511	...

Engagements

EngagementID	EntertainerID	CustomerID	StartDate	EndDate	<<other fields>>
5	1003	10006	2007-09-11	2007-09-14	...
7	1002	10004	2007-09-11	2007-09-18	...
10	1003	10005	2007-09-17	2007-09-26	...
12	1001	10014	2007-09-18	2007-09-26	...

to each Entertainer can be linked multiple rows in Engagements,  
but each Engagement is only linked to a single Entertainer

**Table A**



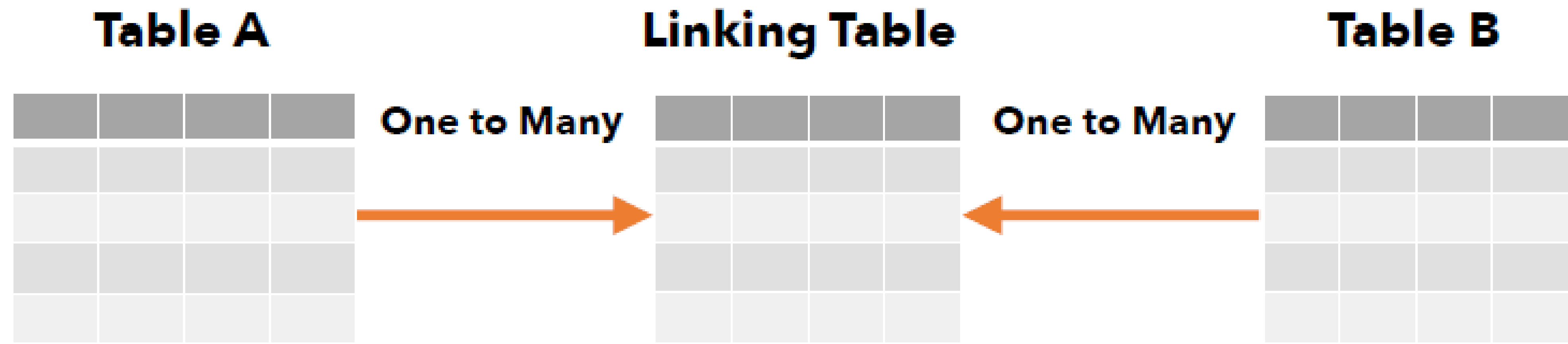
**Many to Many**



**Table B**



Problem: how to create a database structure with a many-to-many relationship type?



## Example:

Customers

CustomerID	CustFirstName	CustLastName	CustPhone	<<other fields>>
10001	Doris	Hartwig	555-2671	...
10002	Deb	Waldal	555-2496	...
10003	Peter	Brehm	555-2501	...

Engagements (linking table)

EngagementID	CustomerID	EntertainerID	StartDate	<<other fields>>
43	10001	1001	2007-10-21	...
58	10001	1002	2007-12-01	...
62	10003	1005	2007-12-09	...
71	10002	1003	2007-12-22	...
125	10001	1003	2008-02-23	...

Entertainers

EntertainerID	EntertainerName	EntertainerPhone	<<other fields>>
1001	Carol Peacock Trio	555-2691	...
1002	Topazz	555-2591	...
1003	JV & the Deep Six	555-2511	...

We create an additional table (linking table) with one to many relationships with the original tables



# Quick quiz

<https://b.socrative.com/login/student/>

Room: SRD2024



Quiz Time

Let's have  
some fun!

# END OF LECTURE 2

## Acreditações e Certificações



UNIGIS



A3ES



Double Degree  
Master Course in  
Information Systems  
Management



Computing  
Accreditation  
Commission

Instituto Superior de Estatística e Gestão de Informação  
Universidade Nova de Lisboa