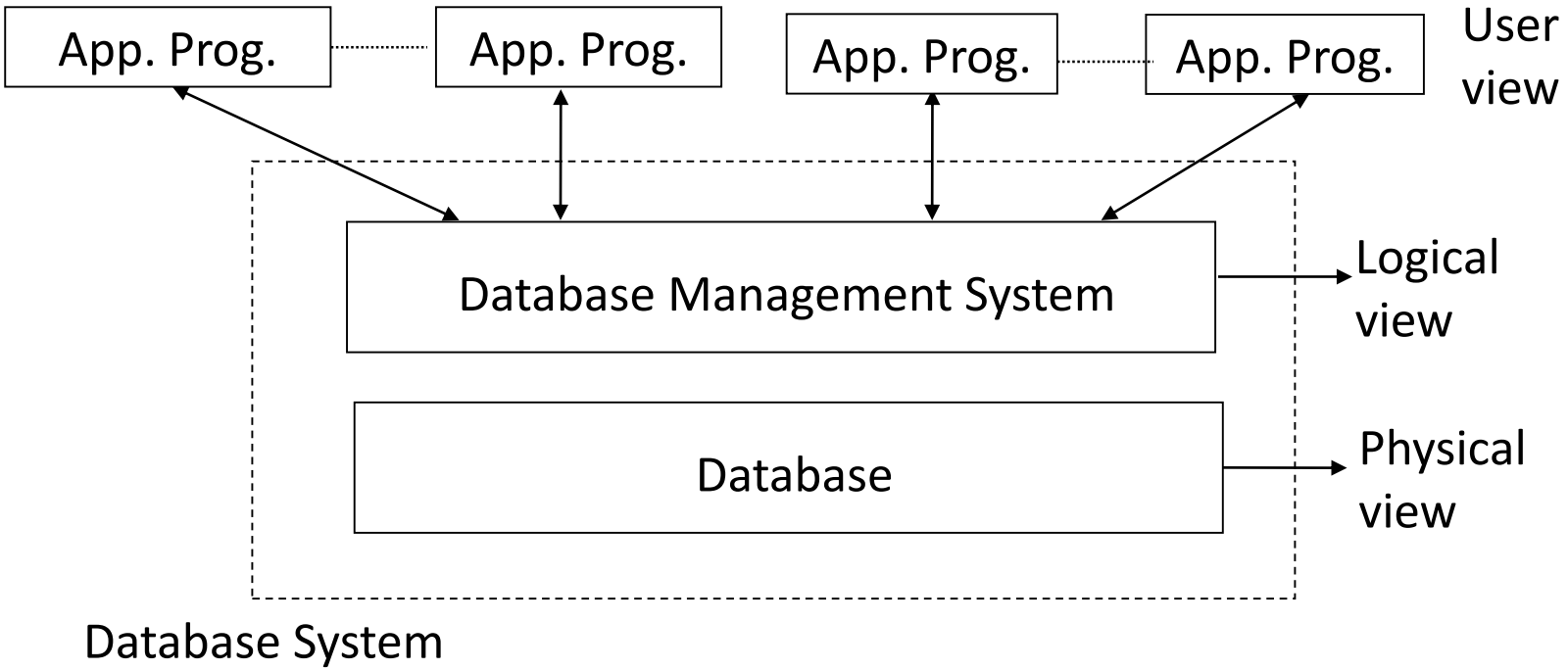
RELATIONAL DATABASES AND SQL

RELATIONAL DATABASES

- Until now we learned:
  - How to obtain an abstract picture of data using E/R diagrams
  - How to determine tables that will be used to store this data
  - This is abstract modeling and design
- Now its time to put this model into action:
  - Creating the tables and filling them with data
  - Accessing data
  - Modifying existing data

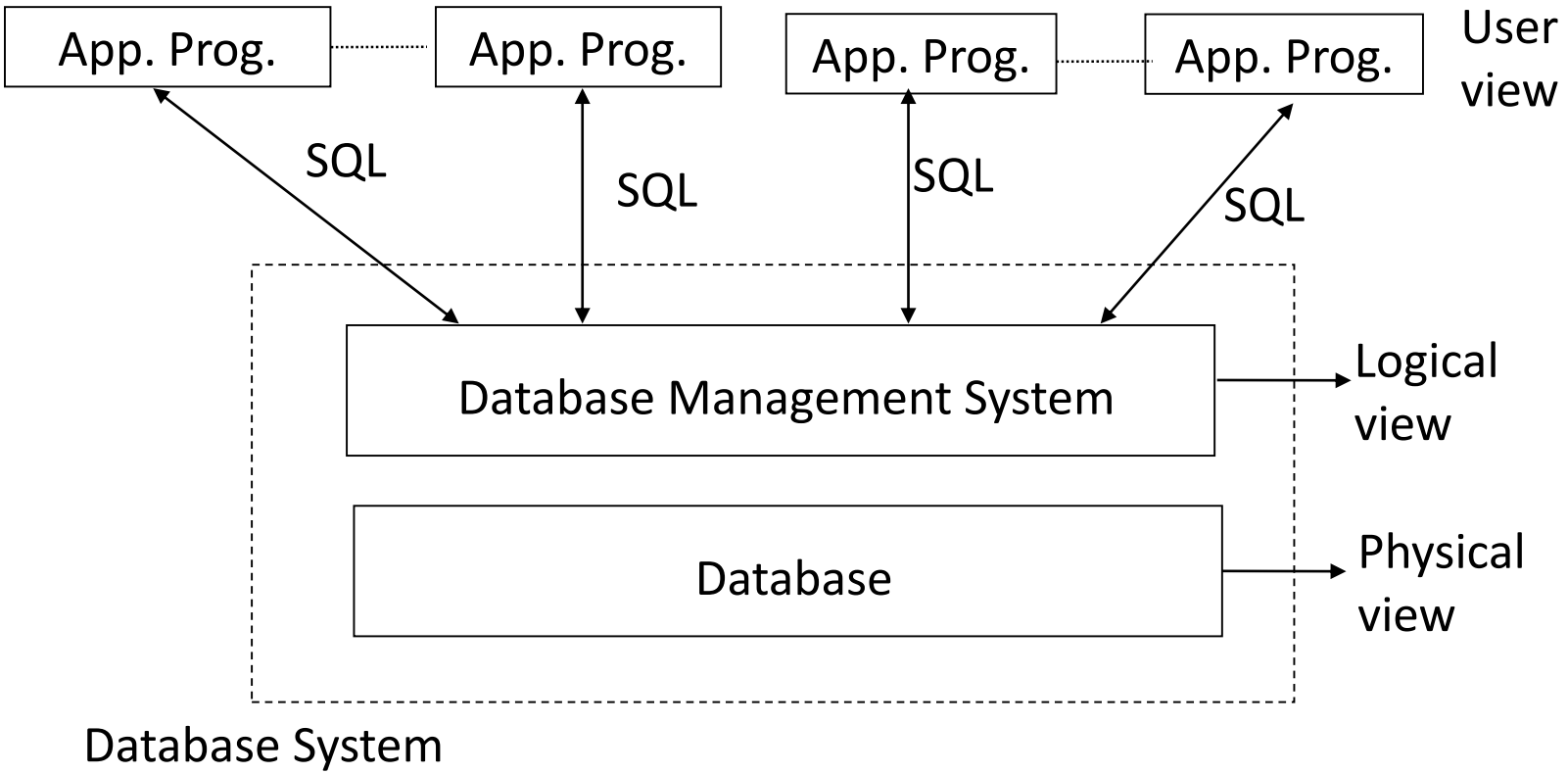
RELATIONAL DATABASES

- We will do this using a database software based on relational model



RELATIONAL DATABASES

- There is a need to communicate with the database
- **Structured query language (SQL)** is a standart language developed for this purpose



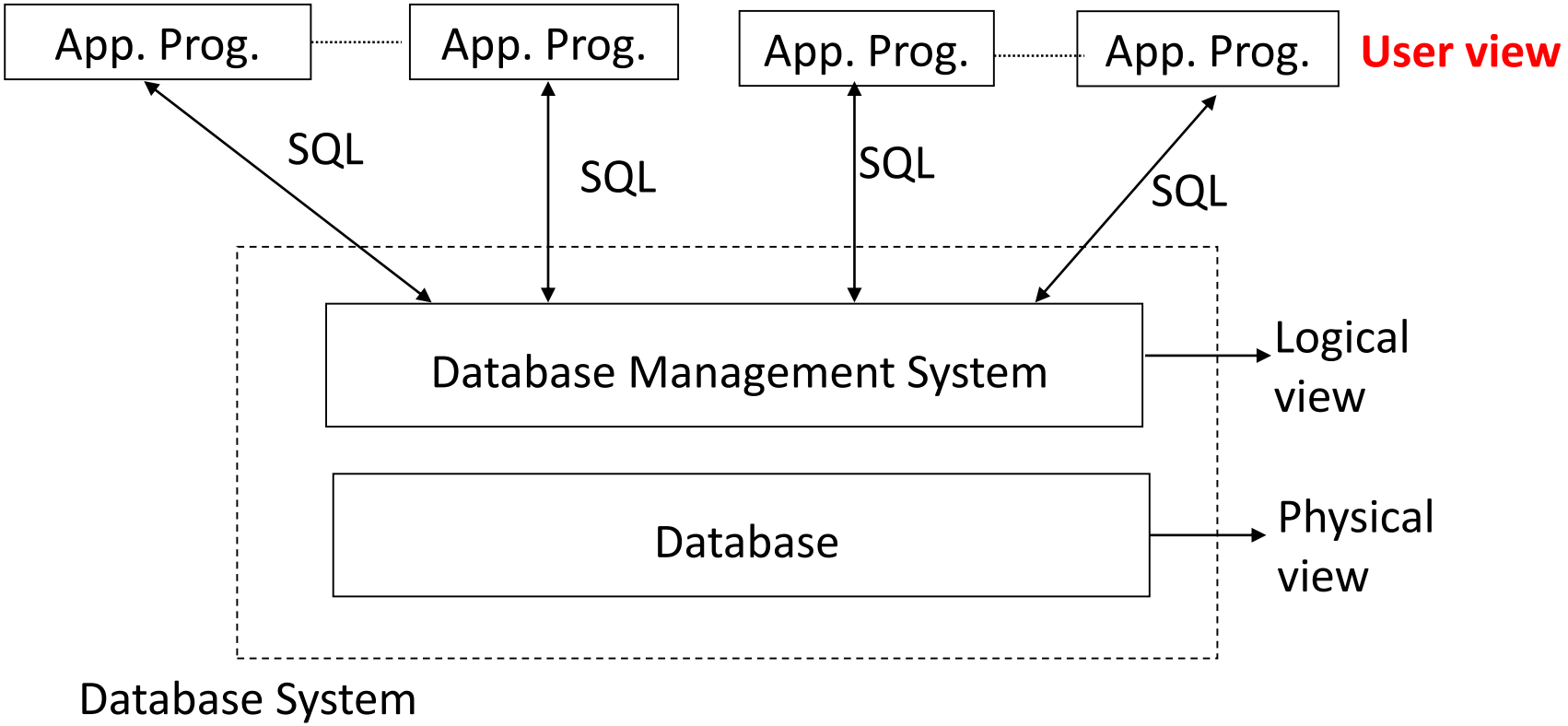
STRUCTURED QUERY LANGUAGE (SQL)

SQL

- SQL is the standard programming language used by relational databases
  - MS Access
  - MS SQL
  - MYSQL (freeware)
  - Oracle
  - Etc.
- However, there are some minor differences in SQL implementations since vendors do not follow the standarts strictly
- In this course, we will use Microsoft SQL Server

RELATIONAL DATABASES

- Since SQL complies with relational model, it is completely based on tables
- That is from user view everything is table



SQL

- Using SQL we can
  - Create/delete tables
  - Enter/modify data
  - Write complex queries to obtain the information we want from the available data
  - Preserve integrity of the data by setting integrity rules
- Table rules set by relational model applies to SQL with few exceptions

TABLE RULES

- Every table must have a name
- Every column of a table must have a name
- In a table:
  - Ordering of columns is not known and will not be used
  - Columns will be referred/identified solely by their names
  - **But this rule is not strictly followed in SQL as we will see**

TABLE RULES

- Columns in different tables:
  - Allowed to have the same name
  - To prevent ambiguity their names will be **qualified** by the table name when necessary
  - Examples:
    - Students.Name
    - Advisors.Name
    - Secretaries.Name

TABLE RULES

- Every element in the same column must be of the same type

COURSES

CID	Name	Credits
SYE346	Management Information Systems	3
SYE222	Operations Research	Four
SYE365	Termodynamics	3
SYE216	Cost Engineering	3
Text	Text	Number

TABLE RULES

- Every row must be different. Duplicates are not allowed

STUDENTS (Entity)

Student ID	Name	Surname	E-mail
20570322	Ali	Ergin	ergin@hotmail.com
20770367	Mehmet	Kavak	mkavak@gmail.com
20850286	Selin	Pekcan	selinp@yahoo.com
20978909	Ercan	Sivri	sivri@gmail.com
20850286	Selin	Pekcan	selinp@yahoo.com

ENROLLMENT (Relation)

Student ID	Course ID
20570322	SYE346
20770367	SYE346
20570322	SYE403
20570322	SYE346

TABLE RULES

- Ordering of rows is not known and will not be used
- Rows of a table will referred/identified by the primary key

STUDENTS

Student ID	Name	Surname	E-mail
20570322	Ali	Ergin	ergin@hotmail.com
20770367	Mehmet	Kavak	mkavak@gmail.com
20850286	Selin	Pekcan	selinp@yahoo.com
20978909	Ercan	Sivri	sivri@gmail.com

- Examples:
  - Student 20570322
  - Student 20850286

FOREIGN KEYS

STUDENTS

Student ID	Name	Surname
20570322	Ali	Ergin
20770367	Mehmet	Kavak
20850286	Selin	Pekcan
20978909	Ercan	Sivri

Primary key

Foreign key: a column which contains only values appearing in a primary column of a table.

ENROLLMENT

Student ID	Course ID
20570322	SYE346
20770367	SYE346
20570322	SYE403
20978909	SYE222
20850286	SYE365

Foreign key

Foreign key

Primary key

COURSES

CID	Name	Credits
SYE346	Management Information Systems	3
SYE222	Operations Research	4
SYE365	Termodynamics	3
SYE216	Cost Engineering	3

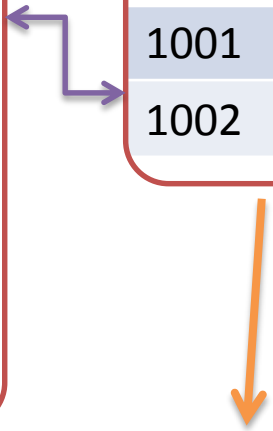
FOREIGN KEYS

STUDENTS

Student ID	Name	Surname	AID
20570322	Ali	Ergin	1001
20770367	Mehmet	Kavak	1002
20850286	Selin	Pekcan	1002
20978909	Ercan	Sivri	1001
20470321	Metin	Yılmaz	1002
20870344	Hasan	Kara	1001

ADVISORS

Advisor ID	Name	Surname
1001	Ugur	Yildiran
1002	Samet	Yılmaz



Primary key

Foreign key

FOREIGN KEYS

- Foreign keys are used to make reference and mainly used for relations
- But it can be used for other purposes as well. For example for multi valued attributes
- A foregin key column is only allowed to contain values currently appearing in the corresponding primary key column

DATA TYPES

DATA TYPES

- We saw that every element of a column must be of the same type
- What are the available data types?
- There are some minor differences on them for different database products
- Let’s have a look to data types in MSSQL Server

DATA TYPES

- Data types for MS SQL Server 2005

Exact numerics

Type	From	To
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal(p,s)	-10^38 +1	10^38 –1
numeric(p,s)	-10^38 +1	10^38 –1
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

DATA TYPES

- Numeric and decimal are fixed precision data types and they are functionally equivalent
- p: the total number of digits used
- s: number of digits to the right of the decimal point
- Example:
  - decimal(5,2) or numeric(5,2) -> 234.32

DATA TYPES

Floating point numbers:

- mantissa x $10^{exponent}$
- Example:
  - 1.245E + 12 = 1.245×10^{12}
  - Mantissa: 1.245
  - Exponent: 12

DATA TYPES

Approximate numerics

Type	From	To
float(n)	-1.79E + 308	1.79E + 308
real	-3.40E + 38	3.40E + 38

- Real does not take any argument
- Argument of float determines the length of the mantissa
  - If 1<=n<=24, mantissa 7 digits
  - If 25<=n<=53, mantissa 15 digits

DATA TYPES

Datetime and smalldatetime

Type	From	To
datetime (3.33 milliseconds accuracy)	Jan 1, 1753	Dec 31, 9999
smalldatetime (1 minute accuracy)	Jan 1, 1900	Jun 6, 2079

- (small) datetime holds both date and time
- If you are interested only with date, set the hour to zero
- There are different formats depending on regional settings
  - ‘dd.mm.yy hh:mm:ss’
  - ‘mm.dd.yy hh:mm:ss’.
  - etc.

DATA TYPES

Character Strings

Type	Description
char(n)	Fixed-length non-Unicode character data with a maximum length of 8,000 characters.
varchar(n)	Variable-length non-Unicode data with a maximum of 8,000 characters.
varchar(max)	Variable-length non-Unicode data with a maximum length of 2^{31} characters (**SQL Server 2005 only**).
text	Variable-length non-Unicode data with a maximum length of 2,147,483,647 characters.

DATA TYPES

- char(n)
  - holds character string of length *n*
  - even if the length of the string is shorter than *n*
  - Example:
    - char(5)
    - Both ‘Ali’ and ‘Ahmet’ occupies 5 bytes
- varchar(n)
  - holds character strings of maximum length *n*
  - Example:
    - varchar(5)
    - ‘Ali’ occupies 3 bytes
    - ‘Ahmet’ occupies 5 bytes
- Comparison:
  - char takes more space but works faster
  - varchar takes less space but works slower

DATA TYPES

- Other data types
  - Unicode Character Strings
  - Binary Strings

SQL STATEMENTS

SQL STATEMENTS

- SQL statements are the comments that we use to communicate with the database
- Using SQL statements we can
  - create/delete tables
  - add/delete rows
  - modify existing data
  - retrieve information
  - ...

SQL STATEMENTS

A brief list of important SQL statements:

DATA MANUPLATION	DATA DEFINITION	ACCESS CONTROL	TRANSACTION CONTROL
SELECT	CREATE TABLE	GRANT	COMMIT
INSERT	DROP TABLE	REVOKE	ROLLBACK
UPDATE	ALTER TABLE	CREATE ROLE	SET TRANSCATION
MERGE	CREATE VIEW	GRANT ROLE	START TRANSACTION
DELETE	DROP VIEW	DROP ROLE	SAVE POINT
…	…	…	…

SQL STATEMENTS

- SQL statements
  - start with a verb
  - contains arguments and expressions
  - not case sensitive
- We will use syntax diagrams to depict syntax of SQL statements

EXAMPLE TABLE

EMPLOYEES

Name	Surname	Birthday	Salary	Tax	Dept	Gender	Address
Ali	Yeşil	10.10.1978 00:00:00	1000	%5	1	M	…. 114/9 Bostancı
Ozan	Kara	05.01.1970 00:00:00	3000	%15	1	M	… 100/3 Bakırköy
Ali	Ergin	03.12.1980 00:00:00	2000	%10	2	M	…. 4/10 Levent ….
Ayşegül	Eren	07.19.1986 00:00:00	2000	%10	4	F	…. 56/2 Bostancı ….
Hasan	Gazi	04.03.1965 00:00:00	1500	%5	3	M	… 20/3 Levent …
Osman	Güneş	02.02.1960 00:00:00	4000	%15	2	M	…. 22/19 Kayışdağı ….
Sezin	Temelli	05.04.1972 00:00:00	2500	%10	3	F	…. 3/2 Üsküdar …

EXAMPLE TABLE

EMPLOYEES

Name	Surname	Birthday	Salary	Tax	Dept	Gender	Address
Ali	Yeşil	10.10.1978 00:00:00	1000	%5	1	M	…. 114/9 Bostancı
Ozan	Kara	05.01.1970 00:00:00	3000	%15	1	M	… 100/3 Bakırköy
Ali	Ergin	03.12.1980 00:00:00	2000	%10	2	M	…. 4/10 Levent ….
Ayşegül	Eren	07.19.1986 00:00:00	2000	%10	4	F	…. 56/2 Bostancı ….
Hasan	Gazi	04.03.1965 00:00:00	1500	%5	3	M	… 20/3 Levent …
Osman	Güneş	02.02.1960 00:00:00	4000	%15	2	M	…. 22/19 Kayışdağı ….
Sezin	Temelli	05.04.1972 00:00:00	2500	%10	3	F	…. 3/2 Üsküdar …

varchar varchar datetime money tinyint tinyint char varchar

CREATING TABLES

- Tables can be created using CREATE TABLE statement
- Basic syntax:
  - CREATE TABLE table-name (column-name data-type, column-name data-type, ….)

CREATING TABLES

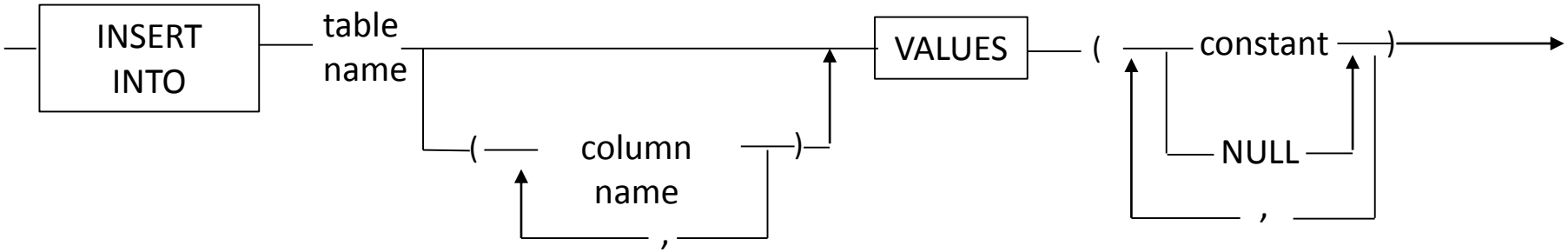
```
CREATE TABLE Employees(
    Name varchar(15),
    Surname varchar(15),
    Birthday datetime,
    Salary money,
    Tax tinyint,
    Dept tinyint,
    Gender char(1),
    Address varchar(50)
    )
```

DELETING TABLES

- Tables can be deleted using DROP TABLE statement
- Example:
  - DROP TABLE Employees

ADDING ROWS

- Rows can be added to a table using INSERT INTO statement
- Syntax diagram:

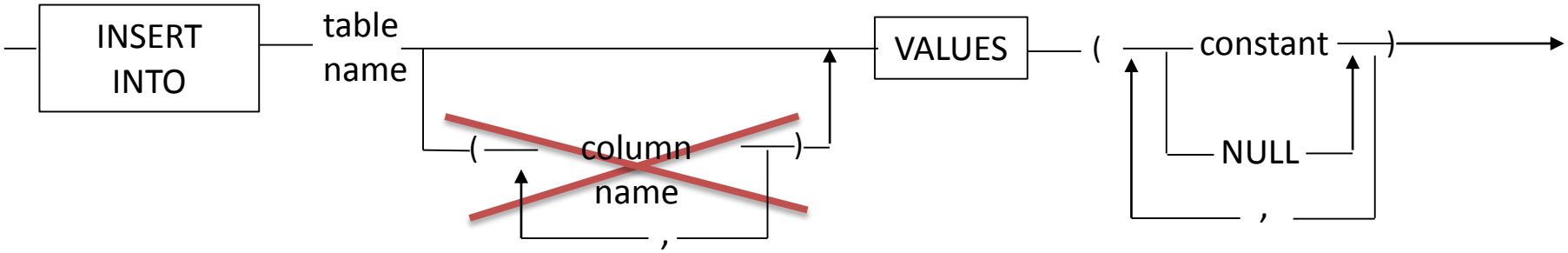


ADDING ROWS

- There are three different ways of adding rows into a table
  - Adding a row by entering values of all columns
  - Adding a row by entering values of some columns
  - Adding multiple rows by using results of a query
- We will study first two at this point

ADDING ROWS

- Entering values of all columns:
  - Skip the first argument and only use the argument starting with VALUES keyword



ADDING ROWS

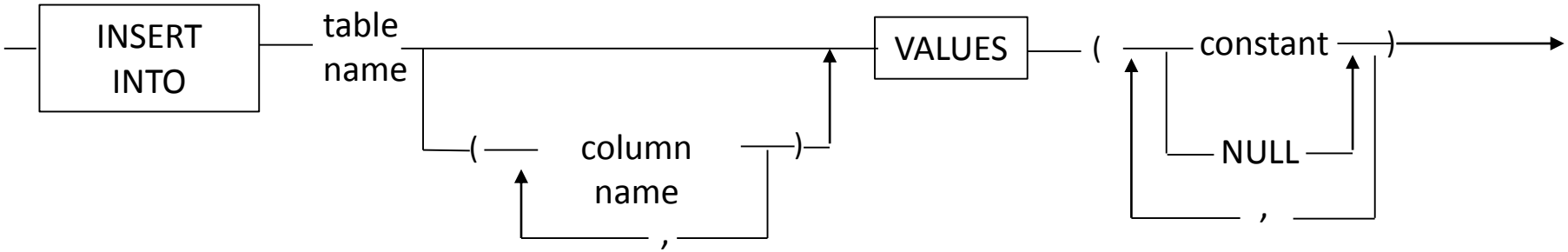
- Values must be entered with the order of columns
- Example:

```
INSERT INTO Employees VALUES ( 'Ozan', 'Kara',
'05.01.1970 00:00:00', 3000, 15, 1, 'M',
'... 100/3, Bakırköy')
```

- This violates the rule of relational model about column access

ADDING ROWS

- Entering values of some columns
  - In this case use the first argument to indicate the names of the columns whose values will be specified



ADDING ROWS

- Order of the column names and values entered must match
- The values of unspecified columns will be set as NULL.
- Example:

  INSERT INTO Employees (Name, Surname, Gender)

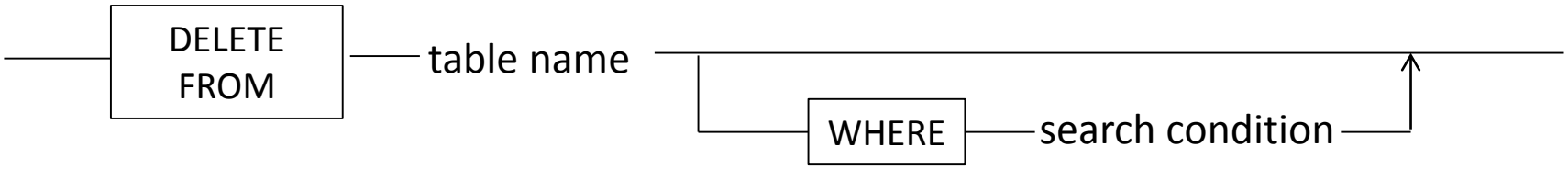
  VALUES ('Mehmet', 'Güneş', 'M')

  – Result:

Mehmet	Güneş	NULL	NULL	NULL	NULL	M	NULL

DELETING ROWS

- Rows of a table can be deleted using DELETE FROM statement
- Syntax:



- If WHERE clause is skipped, all rows will be deleted and the result will be the empty table
- Example:

  DELETE FROM Employees

DELETING ROWS

- If WHERE clause is specified, the rows satifying the *search-condition* expression will be deleted
- Examples:
  - Delete employees working in department 1

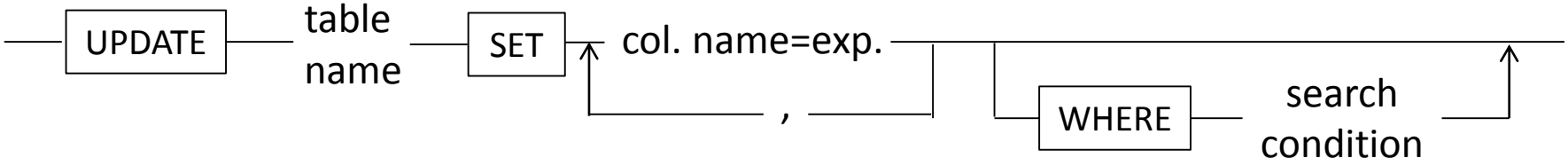
```
DELETE FROM Employees
WHERE dept=1
```

  - Delete employees whose tax is more than 200 TL

```
DELETE FROM Employees
WHERE salary*tax/100>200
```

MODIFYING VALUES

- Values in a table can be modified using UPDATE statement
- Syntax:



MODIFYING VALUES

- Examples:
  - Change Ali Yeşil's department as 4 and make his salary 1500

```
UPDATE Employees
SET Dept=4, Salary=1500
WHERE Name='Ali' AND Surname ='Yeşil'
```

  - Increase salary of all employees by %20

```
UPDATE Employees
SET Salary=Salary*1.2
```

EXPRESSIONS

- Expressions are basically formulas that return a value
  - Examples:
    - salary*tax/100>200
    - Name=’Ali’ AND Surname =’Yeşil’
    - Salary*1.2
- Expressions are composed of
  - constants
  - column names (as variables)
  - operators
  - built-in functions
  - etc.

CONSTANTS

- **Constants:** direct values used in expressions
- There are constants of different data types:
  - Number (integer/decimal)
    - 100, 23.45, etc
  - Floating point number
    - 2E5, 123E-5
  - String
    - ‘istanbul’, ‘Mehmet’
  - Date/time:
    - ’10.01.1980 04:05:10’, ’01.10.1980 04:05:10’

COLUMN NAMES

- Column names can be used as variables in an expression
- Basically a statement scans all rows of a table one by one during execution
- Value of the column name appearing in the expression becomes equal to the value of the cell at the current row
- Example:
  - Salary*1.2

OPERATORS

- Operators:
  - Arithmetic: +,-,/,*,%
    - % is modulus (i.e. reminder from division)
      - Examples: 5%3=2, 11%6=5
    - + is used for both addition and string concatenation
      - Examples: 10+3=13, ‘Mehmet ’+’Güler’=‘Mehmet Güler’
  - Comparison: <,>,<=,<=,<>,!=
  - Logical: AND, OR, NOT
  - etc.
- Examples:
  - salary*tax/100>200
  - Name=’Ali’ AND Surname =’Yeşil’

BUILT-IN FUNCTION

Date functions:

- **GETDATE:** Returns current datetime
  - Example: SELECT GETDATE()
- **YEAR, MOTH, DAY:** These return the corresponding part of datetime as integer.
  - Examples:
    - YEAR('10.01.2010'), result: 2010
    - HOUR(GETDATE()), result: current hour
- **DATEPART:** Returns the specified part of the date as integer.
  - Usage: DATEPART(datepart, date)
  - Example: DATEPART(month,GETDATE()) result: current month

BUILT-IN FUNCTION

- **DATEADD:** Add some interval to date (like 2 days; 1 moth; etc.)
  - Usage: DATEADD(datepart, number, date)
  - Example: DATEADD(month, 3, ’10.01.2010’)
    result: ’10.04.2010’
- **DATEDIFF:** Returns the difference between two dates in terms of a unit (day, month, year) specified as an integer value.
  - Usage: DATEDIFF(datepart, startdate, enddate)
  - Examples:
    - DATEDIFF(day, '10.10.2000', '10.12.2000') result: 61
    - DATEDIFF(month, '10.10.2000', '10.12.2000') result: 2
    - DATEDIFF(year, '10.10.2000', '10.12.2000') result: 0

BUILT-IN FUNCTION

String functions:

- **UPPER, LOWER:** changes the text to upper/lower case
  - Example: UPPER(‘ali’) result: ALİ
- **LEFT, RIGHT:** returns specified number of characters of a string from left/right
  - Example: LEFT(‘Ahmet’, 3) Result: Ahm
- **LEN:** Number of characters in a string
  - Example: LEN(‘Ahmet’) Result: 5