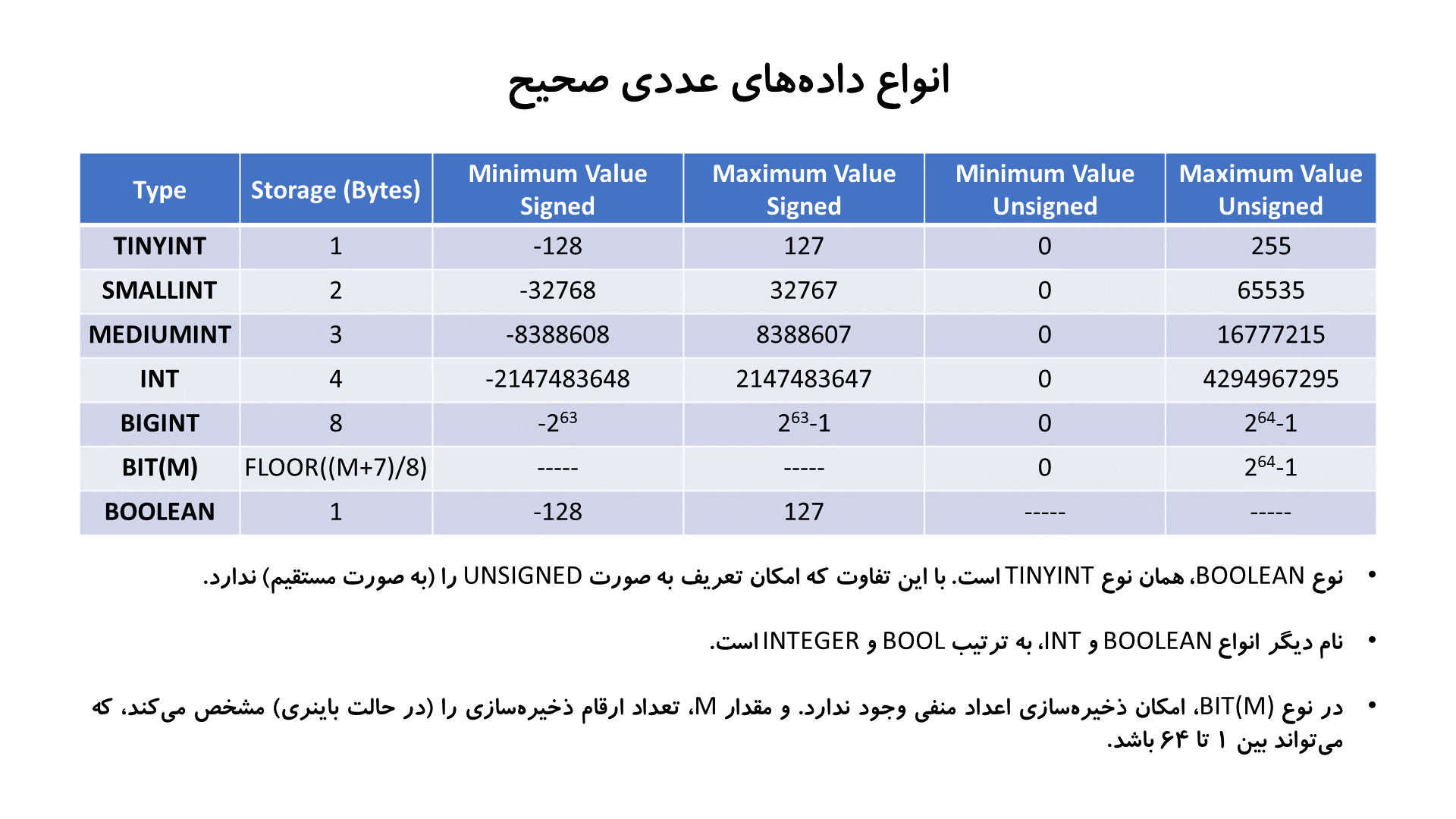
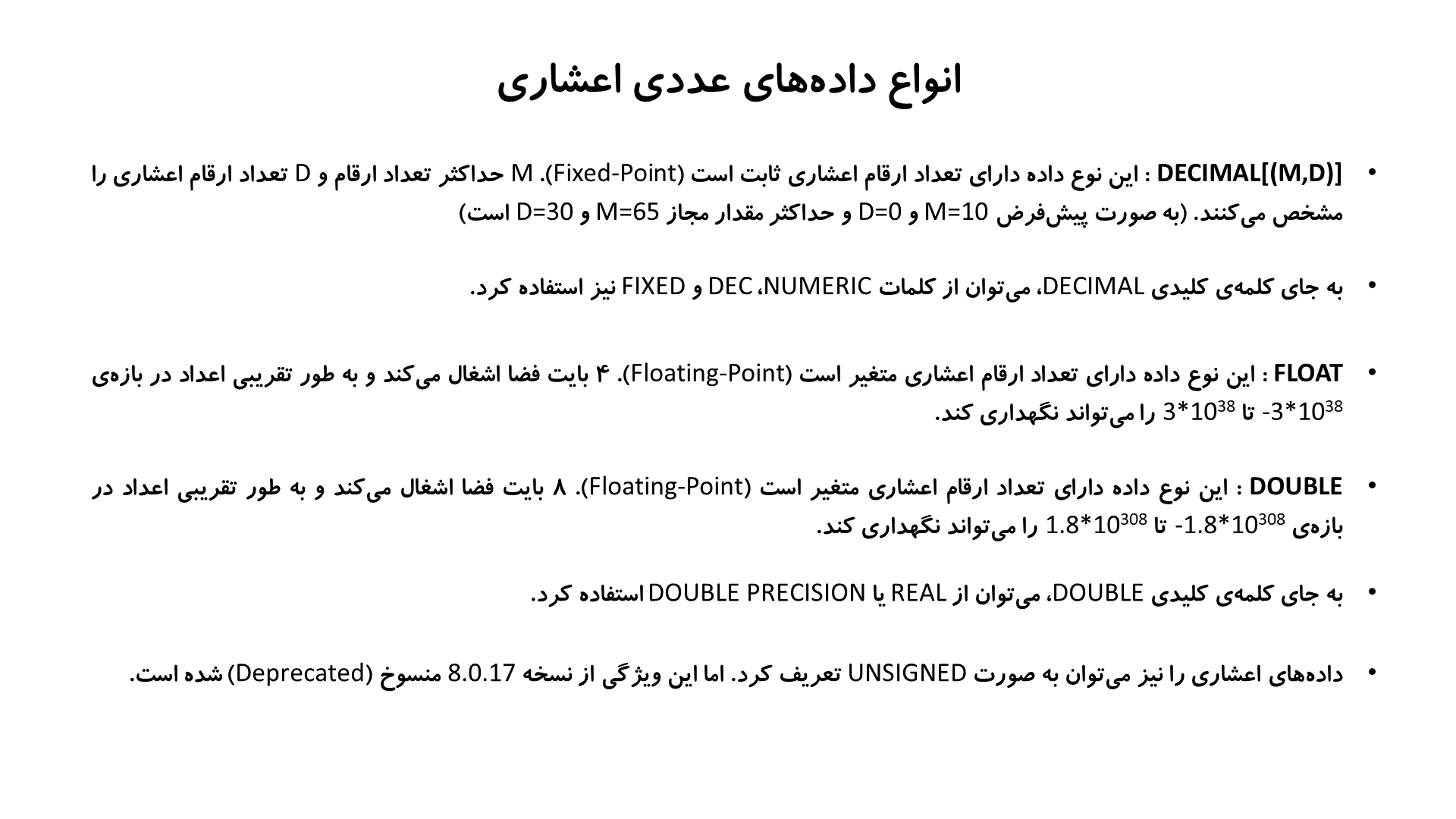
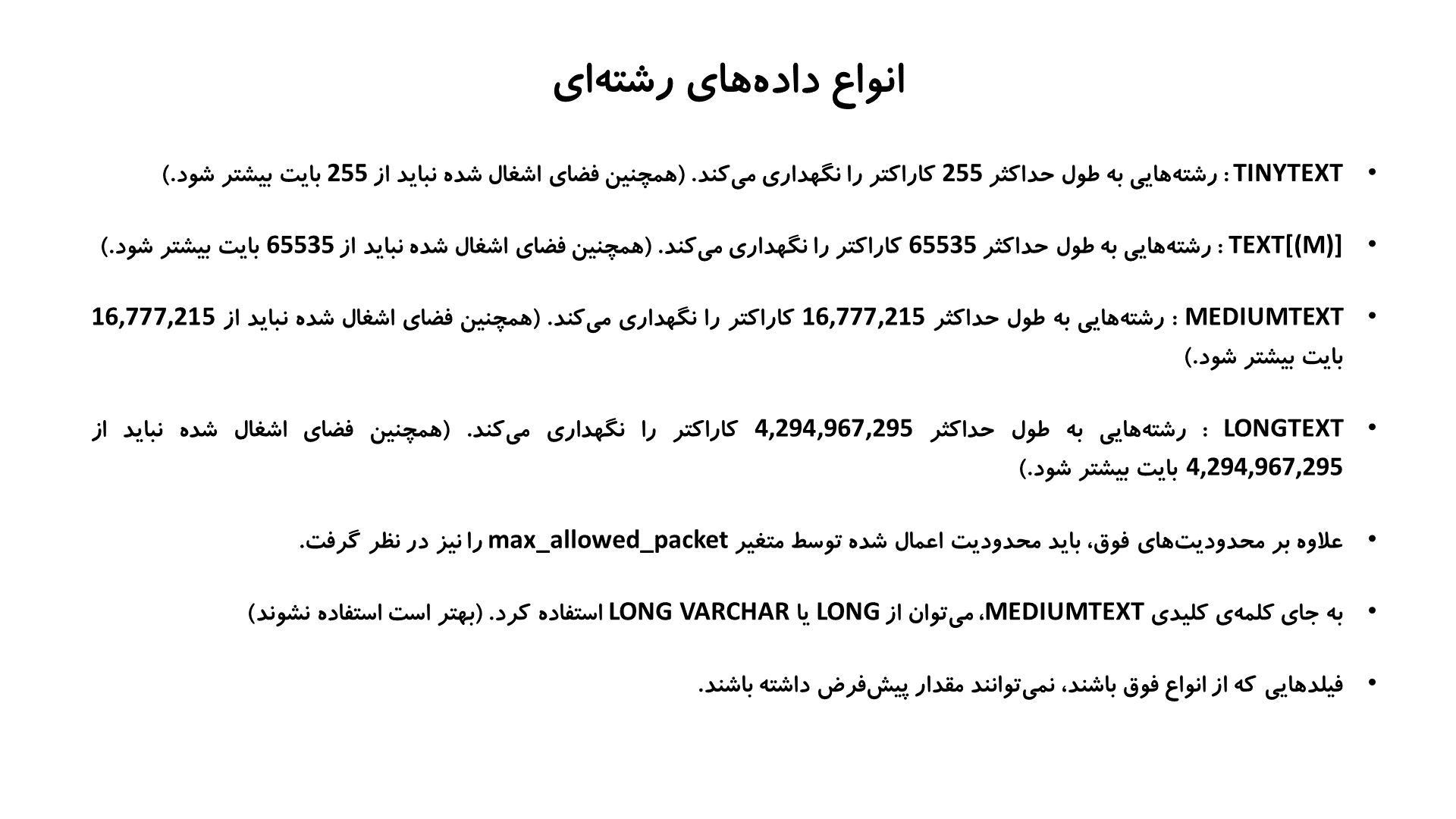
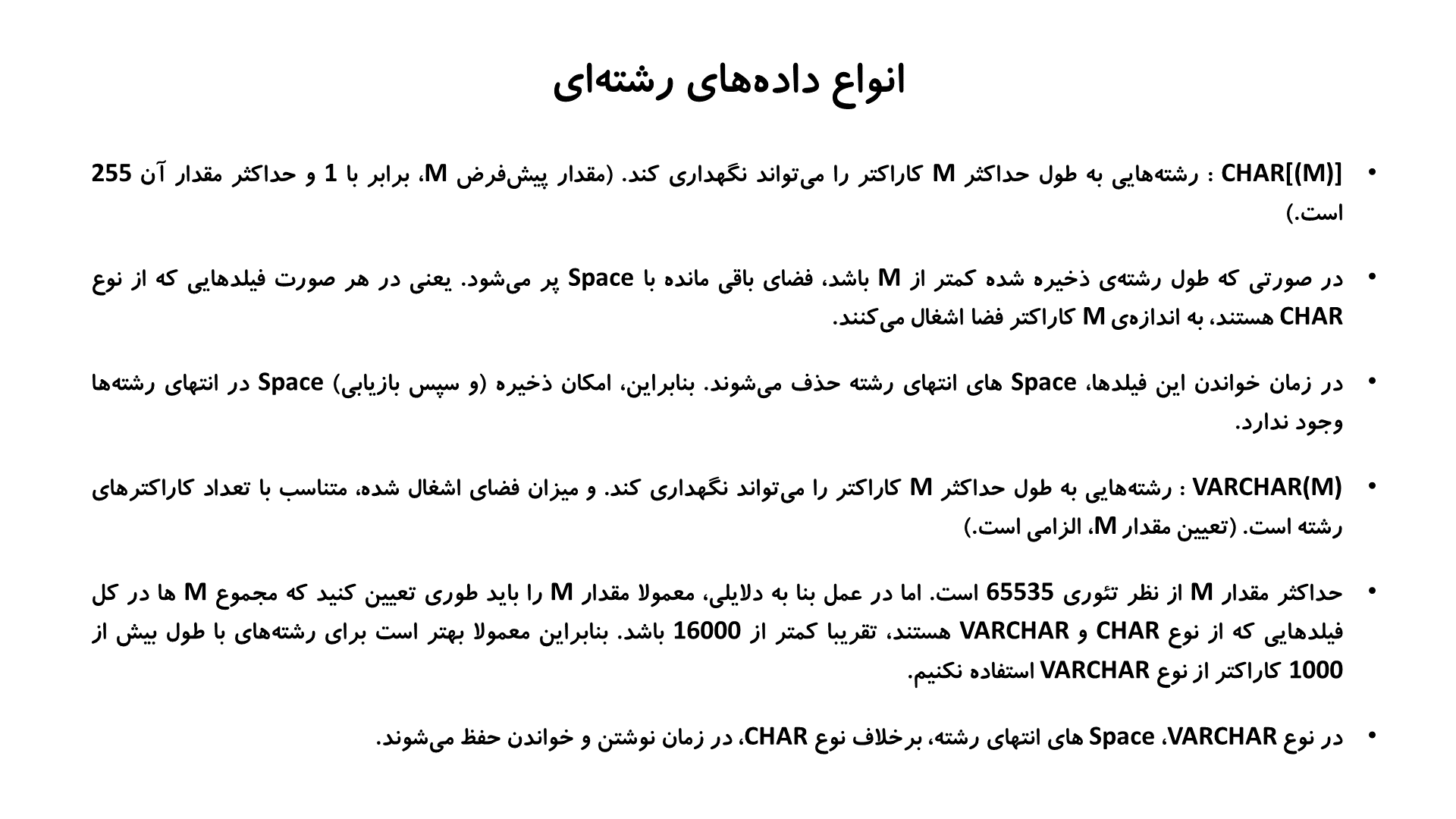
18 – نوع داده عددی :

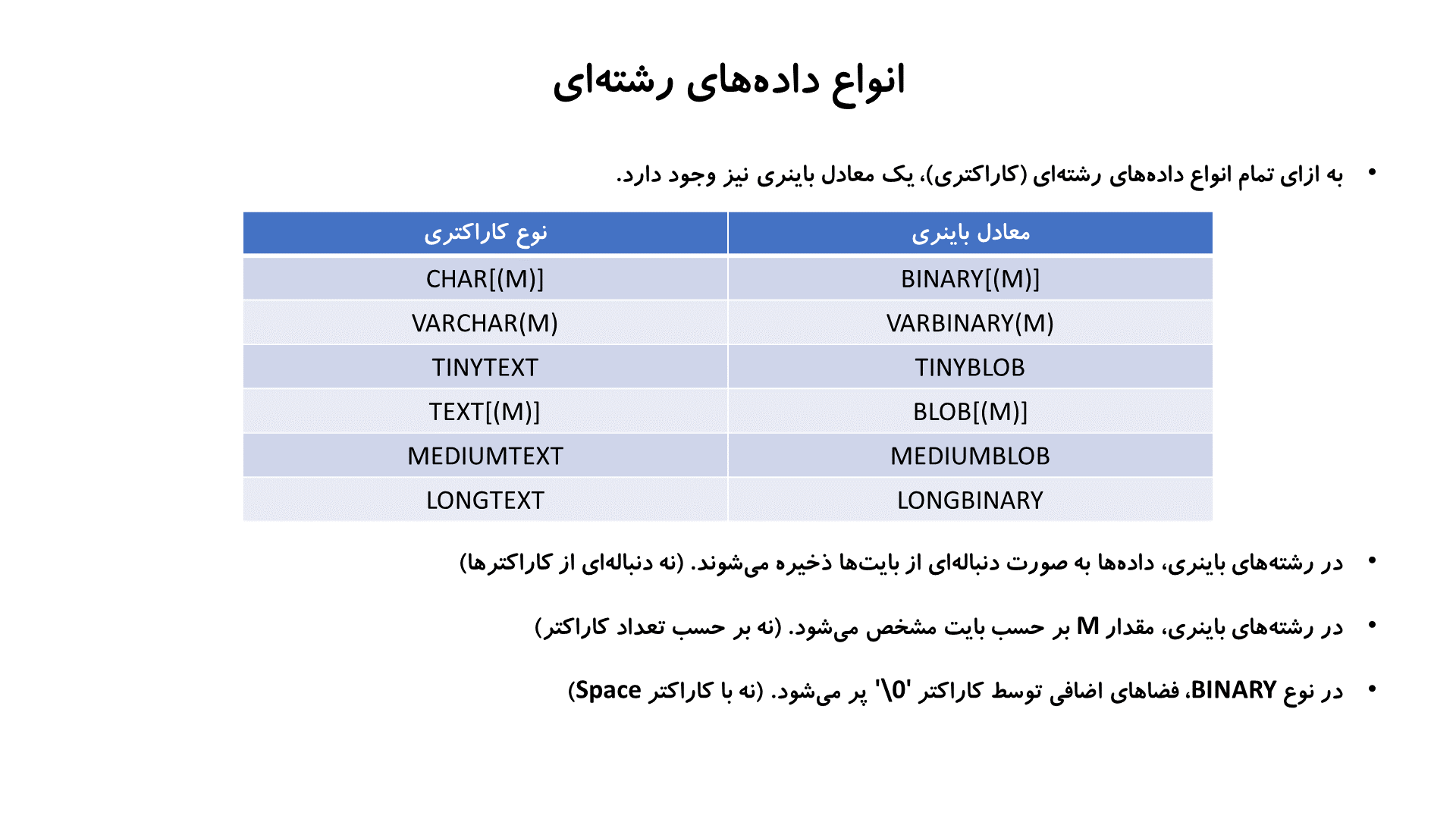
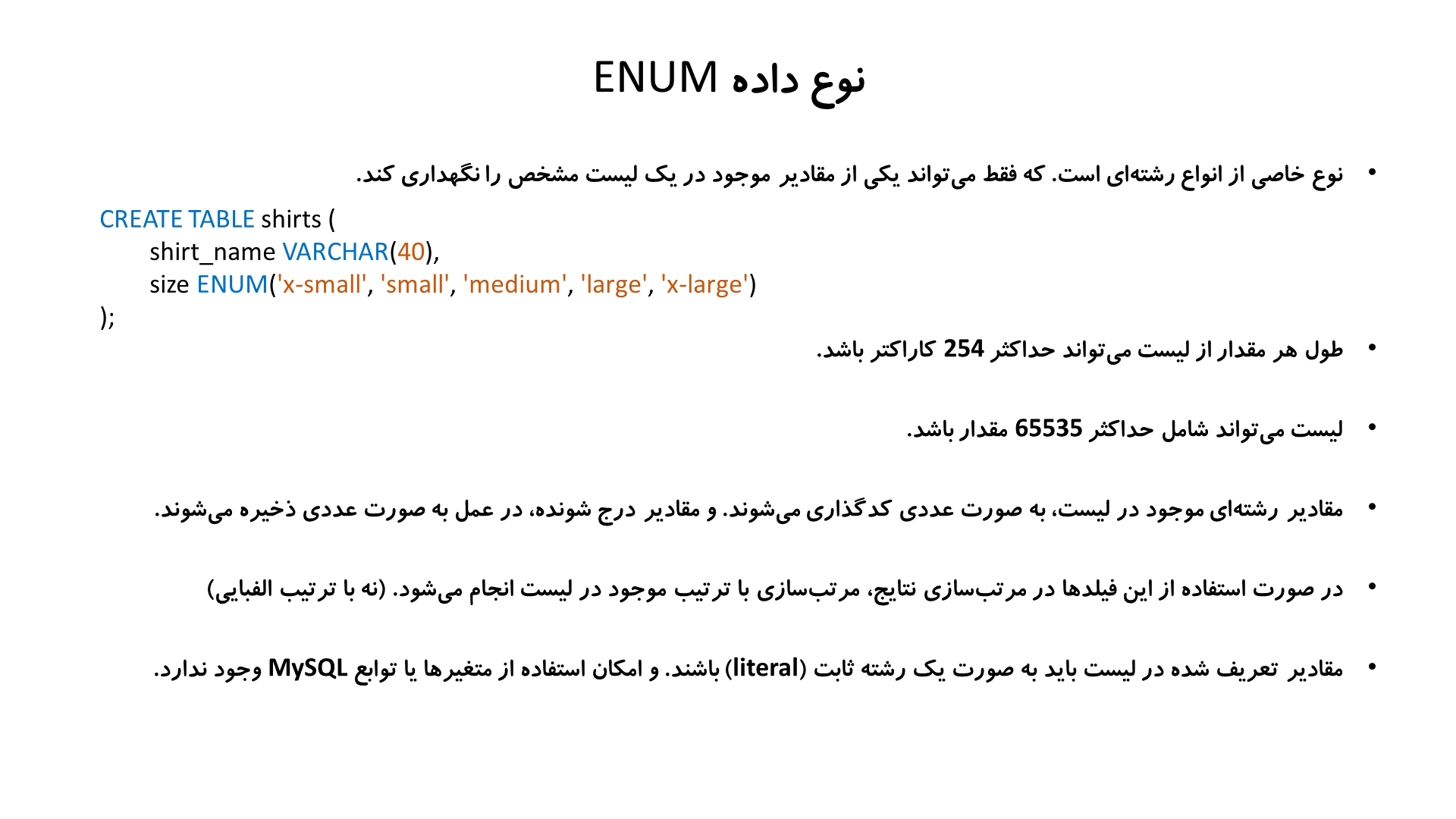


SERIAL : BIGINT UNSIGNED AUTO\_INCREMENT NOT NULL UNIQUE

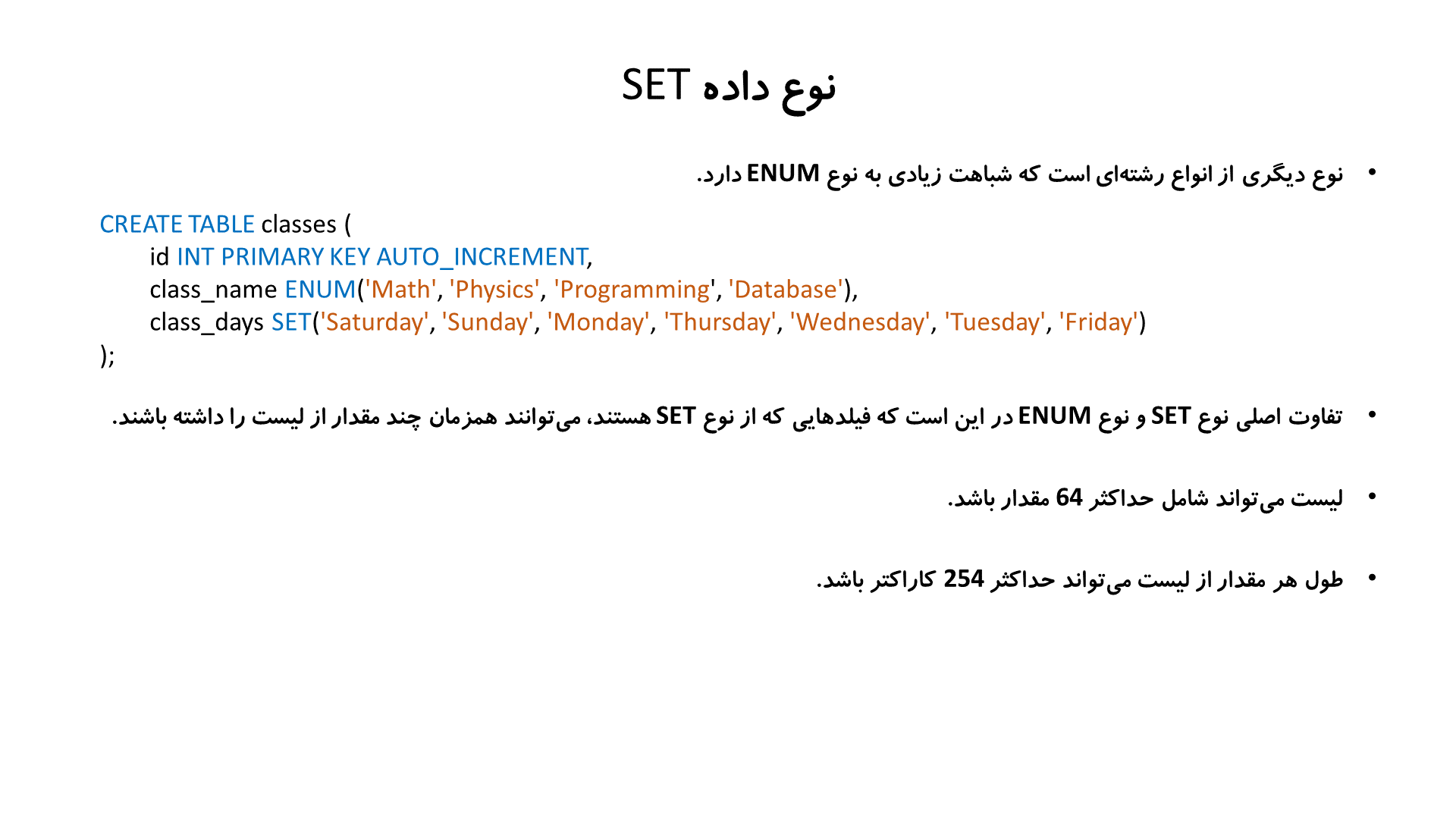


19 – string data



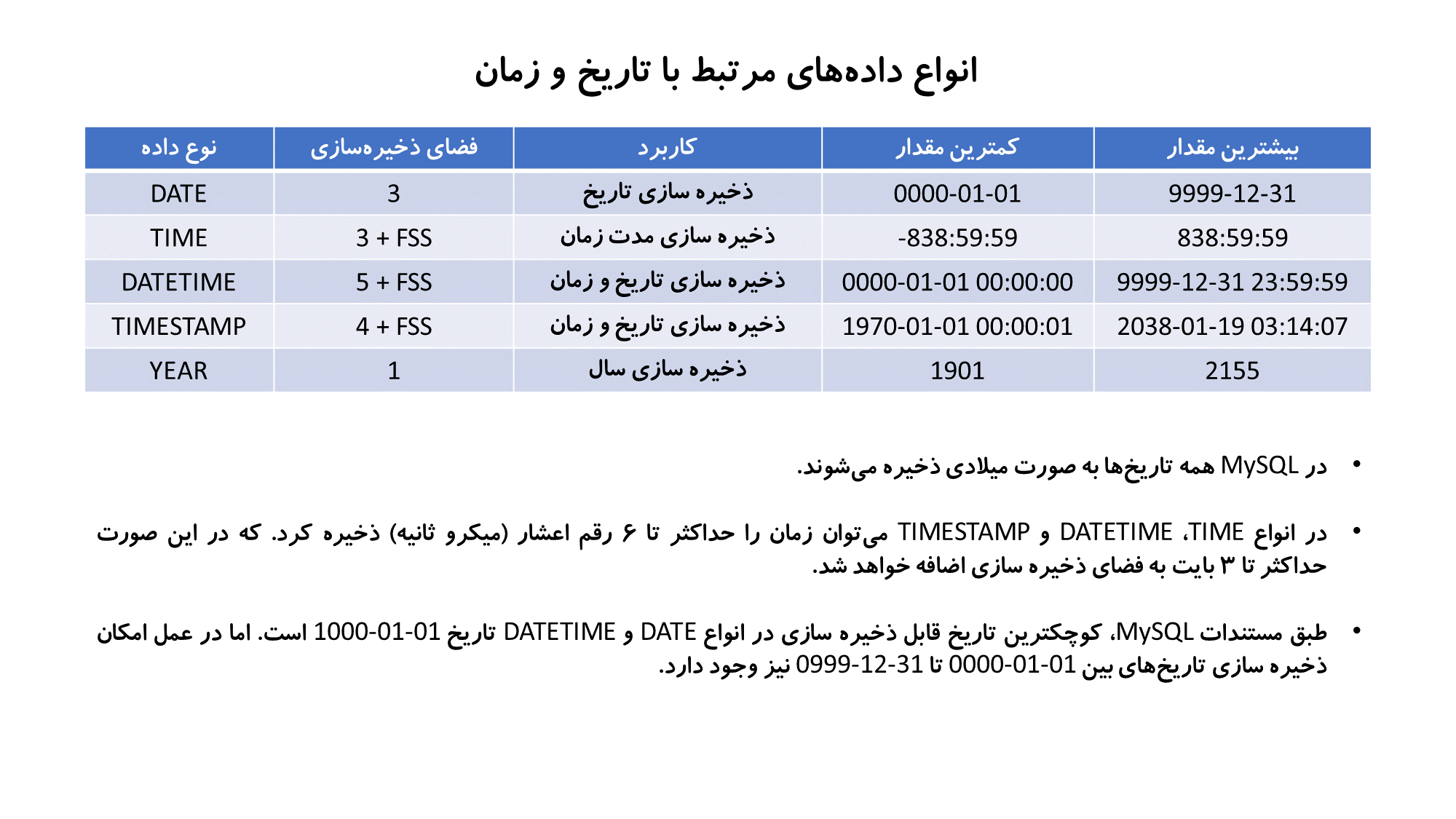
20 – SET و 

select \* from sirt order by CAST(size as char) ;

 select \* from sirt where FIELD\_IN\_SET('کارآموزی', wishList) اونایی رو که کارآموزی رو دارن نشون بده

select \* from sirt where wishList= ‘کارآموزی,ارشد’ اونایی که دقیقا ارشد و کارآموزی رو دارن بده (نه کمترنه بیشتر )

21 – داده های تاریخ و زمان

  
create table datetimes(

id serial PRIMARY key ,

d1 DATETIME,

d2 datetime(3) default '2023-06-23 21:23:11.345',

d3 datetime default current\_timestamp,

d4 datetime on update current\_timestamp,

other INT default 0

);

INSERT INTO datetimes(datetime1,datetime2) VALUES('1222-11-12 20:11:44','2023-10-22T22:11:34.3412');

/\*

تاریخ و زمان را می‌توان با فاصله یا کاراکتر

T

از یکدیگر جدا کرد

\*/

INSERT INTO datetimes(datetime1,datetime2) VALUES('2000-10-11' , '16-11-27');

-- در صورت مشخص نکردن مقدار زمان، مقدار 00:00:00 برای آن در نظر گرفته می‌شود

INSERT INTO datetimes(datetime1) VALUES('10:20:30'); -- Error

-- امکان درج زمان بدون تاریخ وجود ندارد

Timestamp :

DROP TABLE IF EXISTS timestamps;

CREATE TABLE timestamps(

id INT PRIMARY KEY AUTO\_INCREMENT,

timestamp1 TIMESTAMP,

timestamp2 TIMESTAMP(3) DEFAULT '2023-06-23 22:11:33.443',

timestamp3 TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

timestamp4 TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

other\_field INT DEFAULT 0

);

INSERT INTO timestamps(timestamp1) VALUES('1970-01-01 00:00:01' ); -- Error

/\*

در نوع داده

TIMESTAMP

زمان همیشه بر مبنای زمان جهانی ذخیره می‌شود

\*/

INSERT INTO timestamps(timestamp1) VALUES('1970-01-01 00:00:01+00:00');

22 – constraints

انواع محدودیت ها : column level – table level

create table enroles(

id serial primary key,

student\_id int not null,

course\_id int not null,

grade DECIMAL(4, 2) check(grade >= 0 and grade <=20)

-- or

-- grade DECIMAL(4, 2) constraint const\_name check(grade >= 0 and grade <=20)

);

در جدول بالا فقط از محدودیت column level استفاده شده .

Table level : میتونیم رو ترکیب چند تا ستون محدودیت بزاریم .

create table enroles(

id serial primary key,

student\_id int not null,

course\_id int not null,

grade DECIMAL(4, 2) check(grade >= 0 and grade <=20)

constraint check\_1 check(grade >= 0 and grade <=20 and course\_id >= 100)

)

create table enroles(

id serial primary key,

student\_id int not null,

course\_id int not null,

grade DECIMAL(4, 2) check(grade >= 0 and grade <=20),

constraint unq\_key UNIQUE KEY(student\_id, course\_id)

-- or

-- UNIQUE(student\_id, course\_id)

-- or

-- UNIQUE unq\_key (student\_id, course\_id)

)

انواع محدودیت ها :

Column constraints include:

* NOT NULL

Specifies that this column cannot hold NULL values (constraints of this type are not nameable).

* PRIMARY KEY

Specifies the column that uniquely identifies a row in the table. The identified columns must be defined as NOT NULL.

**Note:**If you attempt to add a primary key using ALTER TABLE and any of the columns included in the primary key contain null values, an error will be generated and the primary key will not be added. See [ALTER TABLE statement](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqlj81859.html#rrefsqlj81859) for more information.

* UNIQUE

Specifies that values in the column must be unique.

* FOREIGN KEY

Specifies that the values in the column must correspond to values in a referenced primary key or unique key column or that they are NULL.

* CHECK

Specifies rules for values in the column.

Table constraints include:

* PRIMARY KEY

Specifies the column or columns that uniquely identify a row in the table. NULL values are not allowed.

* UNIQUE

Specifies that values in the columns must be unique.

* FOREIGN KEY

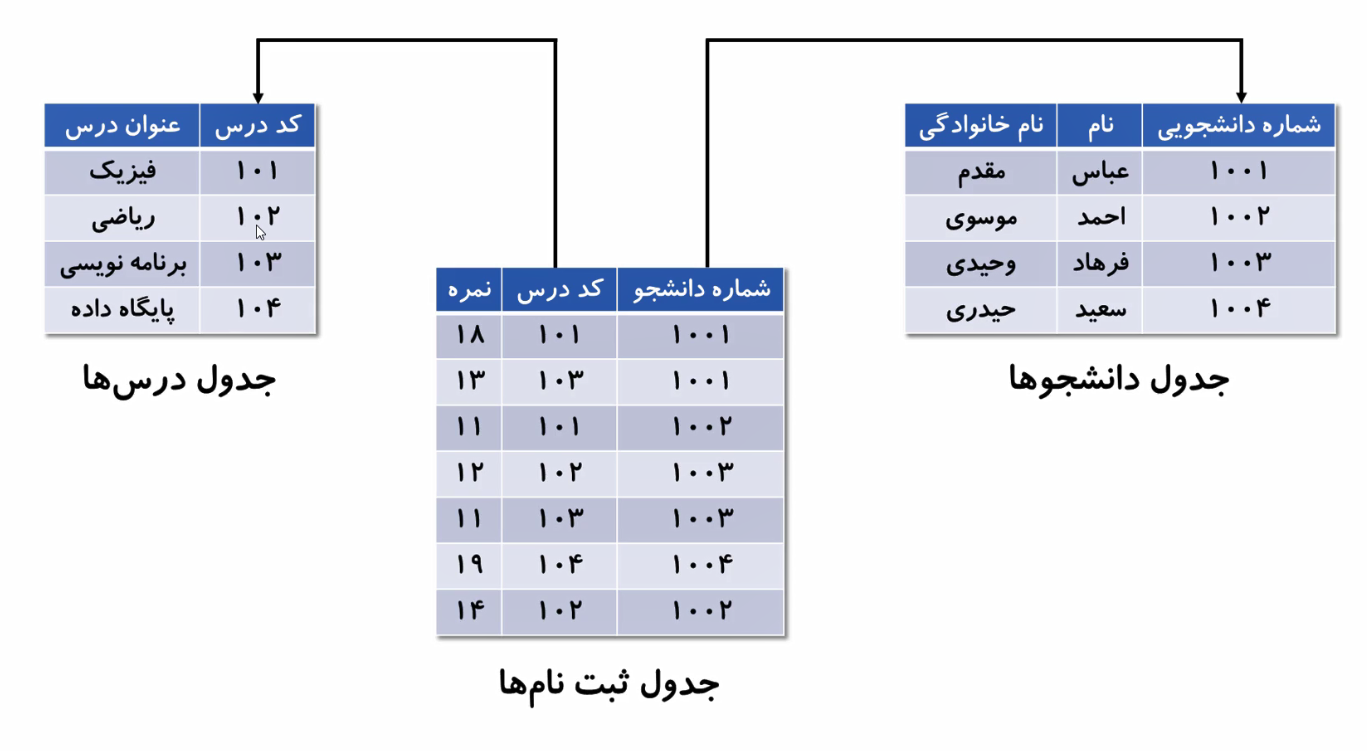
Specifies that the values in the columns must correspond to values in referenced primary key or unique columns or that they are NULL.

**Note:**If the foreign key consists of multiple columns, and *any* column is NULL, the whole key is considered NULL. The insert is permitted no matter what is on the non-null columns.

* CHECK

Specifies a wide range of rules for values in the table.

23 – Foreign Key



**drop table if exists students;**

**create table students(**

**id serial primary key,**

**first\_name varchar(40),**

**last\_name varchar(50)**

**);**

**insert into students(first\_name, last\_name) values**

**('عباس', 'مقدم'), ('احمد', 'موسوی'),**

**('فرهاد', 'وحیدی'), ('سعید', 'حیدری');**

**drop table if exists courses;**

**create table courses(**

**id serial primary key,**

**course\_name varchar(50) not null**

**);**

**insert into courses(course\_name) values ('فیزیک'),( 'ریاضی'),( 'برنامه نویسی'),( 'پایگاه داده'),( 'شیمی'),( 'الگوریتم');**

**create table enrols(**

**id serial,**

**student\_id int not null,**

**course\_id int not null,**

**grade decimal(4, 2) check(grade >= 0 and grade <= 20),**

**primary key(id),**

**constraint unq\_std\_id\_crs\_id unique (student\_id, course\_id),**

**constraint enrols\_students\_fk foreign key(student\_id) references students(id)**

**on delete cascade on update cascade,**

**constraint enrols\_courses\_fk foreign key(course\_id) references courses(id)**

**on delete set null on update cascade**

**-- or**

**-- foreign key(student\_id) references students(id),**

**-- enrols\_courses\_fk foreign key(course\_id) references courses(id)**

**);**

**-- available actions : cascade, set null, restrict, no action**

**فارن کی در خوده جدول :**

**create table categories(**

**id int primary key auto\_increment,**

**cat\_name varchar(5) not null,**

**parent\_id int,**

**foreign key(parent\_id) references categories(id) on delete cascade**

**);**

**24 – alter database and table**

**Alter database db\_name default character set utf8mb4 default collate ….**

**Alter database db\_name rename**

**ALTER TABLE courses ADD COLUMN teacher VARCHAR(40);**

**ALTER TABLE courses ADD COLUMN teacher2 VARCHAR(40) FIRST;**

**ALTER TABLE courses ADD COLUMN teacher3 VARCHAR(40) AFTER course\_id;**

**ALTER TABLE courses ADD COLUMN teacher4 VARCHAR(40) NOT NULL DEFAULT 'a' AFTER teacher3;**

**ALTER TABLE courses RENAME COLUMN course\_name TO course\_title;**

**ALTER TABLE courses CHANGE COLUMN course\_title course\_name VARCHAR(80);**

**ALTER TABLE courses CHANGE COLUMN course\_id course\_id BIGINT;**

**ALTER TABLE courses CHANGE COLUMN course\_id course\_id SMALLINT;**

**ALTER TABLE courses CHANGE COLUMN course\_id course\_id VARCHAR(20);**

**-- در صورت تغییر نوع داده، باید نوع داده‌ی جدید، با داده‌های موجود سازگار باشد**

**ALTER TABLE courses CHANGE COLUMN course\_id course\_id TIMESTAMP; -- Error**

**ALTER TABLE courses CHANGE COLUMN course\_name course\_name VARCHAR(10); -- Error**

**ALTER TABLE courses CHANGE COLUMN course\_id course\_id INT AFTER course\_name;**

**ALTER TABLE courses RENAME TO course;**

**RENAME TABLE course TO courses;**

**ALTER TABLE courses DEFAULT CHARSET latin1 ENGINE MyISAM;**

**ALTER TABLE courses**

**ADD COLUMN teacher1 VARCHAR(40) ,**

**RENAME COLUMN course\_name TO course\_title ,**

**ADD COLUMN teacher2 VARCHAR(40) AFTER course\_id;**

**-- می‌توان همزمان چند تغییر در جدول ایجاد کرد. برای این منظور باید تغییرات مورد نظر را با کاما از هم جدا کرد**

**RENAME TABLE university.courses TO mydb.courses2;**

**/\***

**با استفاده از دستور**

**RENAME TABLE**

**یا دستور**

**ALTER TABLE ... RENAME**

**می‌توان جدول‌ها را از یک دیتابیس به دیتابیس دیگر منتقل کرد**

**\*/**

**ALTER TABLE students DROP COLUMN student\_id;**

**ALTER TABLE courses DROP PRIMARY KEY; -- Error**

**ALTER TABLE courses CHANGE COLUMN course\_id course\_id INT;**

**ALTER TABLE courses DROP PRIMARY KEY;**

**ALTER TABLE courses ADD PRIMARY KEY(course\_id);**

**ALTER TABLE courses CHANGE COLUMN course\_id course\_id INT AUTO\_INCREMENT;**

**ALTER TABLE courses ADD CONSTRAINT unq UNIQUE(course\_name);**

**ALTER TABLE courses DROP CONSTRAINT unq;**

**ALTER TABLE enrols DROP CONSTRAINT chk; -- Or, DROP CHECK chk**

**ALTER TABLE enrols ADD CONSTRAINT chk CHECK(grade >= 0 AND grade <= 20);**

**ALTER TABLE enrols ADD CONSTRAINT fk FOREIGN KEY(course\_id) REFERENCES courses(course\_id);**

**ALTER TABLE enrols DROP CONSTRAINT fk;**

**52 – 26 – join**

**Cross Join : ضرب دکارتی**

**تمامی ترکیب های دوتایی بین دوتا جدول**

**SELECT \* FROM enrols CROSS JOIN courses;**

**SELECT \* FROM courses CROSS JOIN enrols;**

**SELECT \* FROM courses CROSS JOIN enrols ORDER BY course\_name;**

**SELECT \* FROM courses CROSS JOIN enrols ORDER BY course\_id; -- Error**

**SELECT \* FROM courses CROSS JOIN enrols ORDER BY courses.course\_id;**

**SELECT \* FROM courses CROSS JOIN enrols ORDER BY enrols.course\_id;**

**SELECT \* FROM courses , enrols;**

**SELECT \* FROM courses , enrols**

**WHERE courses.course\_id = enrols.course\_id ORDER BY courses.course\_id;**

**SELECT courses.\* , enrols.student\_id , enrols.grade FROM courses , enrols**

**WHERE courses.course\_id = enrols.course\_id ORDER BY courses.course\_id;**

**SELECT \* FROM students , courses , enrols;**

**SELECT students.firstname , students.lastname , courses.course\_name , enrols.grade**

**FROM students , courses , enrols**

**WHERE students.student\_id = enrols.student\_id AND courses.course\_id = enrols.course\_id;**

**SELECT CONCAT(students.firstname , ' ', students.lastname) AS fullname ,**

**courses.course\_name , enrols.grade**

**FROM students , courses , enrols**

**WHERE students.student\_id = enrols.student\_id AND courses.course\_id = enrols.course\_id**

**ORDER BY grade DESC;**

**SELECT CONCAT(students.firstname , ' ', students.lastname) AS fullname ,**

**courses.course\_name , enrols.grade**

**FROM students , courses , enrols**

**WHERE students.student\_id = enrols.student\_id AND courses.course\_id = enrols.course\_id**

**AND grade > 15 ORDER BY grade DESC;**

**/\***

**معمولا از کلیدهای خارجی به عنوان شرط اتصال جدولها استفاده می شود.**

**اما هیچ الزامی برای این کار وجود ندارد و از هر فیلدی میتوان برای اتصال جدولها استفاده کرد**

**\*/**

**Inner Join :**

**SELECT \* FROM students INNER JOIN enrols;**

**SELECT \* FROM students INNER JOIN enrols ON students.student\_id = enrols.student\_id;**

**SELECT \* FROM students INNER JOIN courses INNER JOIN enrols**

**ON students.student\_id = enrols.student\_id AND courses.course\_id = enrols.course\_id;**

**SELECT \* FROM students INNER JOIN courses INNER JOIN enrols**

**ON students.student\_id = enrols.student\_id AND courses.course\_id = enrols.course\_id**

**WHERE grade > 15 ORDER BY grade DESC;**

**SELECT CONCAT(students.firstname , ' ', students.lastname) AS fullname ,**

**courses.course\_name , enrols.grade**

**FROM students INNER JOIN courses INNER JOIN enrols**

**ON students.student\_id = enrols.student\_id AND courses.course\_id = enrols.course\_id**

**WHERE grade > 15 ORDER BY grade DESC;**

**/\***

**در صورتی که بین نام جدولها به جای**

**INNER JOIN**

**از کاما استفاده کنیم و همچنین شرط اتصال را هم به جای**

**ON**

**با استفاده از**

**WHERE**

**بیان کنیم.**

**INNER JOIN**

**به**

**CROSS JOIN**

**تبدیل می‌شود. در واقع می‌توان گفت**

**INNER JOIN**

**حالت خاصی از**

**CROSS JOIN**

**است**

**\*/**

**/\***

**در صورتی که نام فیلدهای مورد استفاده برای اتصال جدولها، در هر دو جدول یکی باشد، میتوان با استفاده از**

**USING**

**شرط اتصال را به صورت خلاصه بیان کرد.**

**\*/**

**SELECT \* FROM courses INNER JOIN enrols USING(course\_id);**

**/\***

**در دستور بالا نمی‌توان از کاما به جای**

**INNER JOIN**

**استفاده کرد**

**\*/**

**SELECT \* FROM courses NATURAL JOIN enrols;**

**/\***

**ستون‌های همنام در**

**NATURAL JOIN**

**فقط یک بار ظاهر می‌شوند**

**\*/**

**SELECT CONCAT(students.firstname , ' ', students.lastname) AS fullname ,**

**courses.course\_name , enrols.grade**

**FROM students INNER JOIN courses INNER JOIN enrols**

**USING(student\_id , course\_id)**

**WHERE grade > 15 ORDER BY grade DESC;**

**SELECT CONCAT(students.firstname , ' ', students.lastname) AS fullname ,**

**courses.course\_name , enrols.grade**

**FROM students NATURAL JOIN courses NATURAL JOIN enrols**

**WHERE grade > 15 ORDER BY grade DESC;**

**/\***

**در صورت استفاده همزمان از**

**INNER JOIN و NATURAL JOIN**

**ترتیب جدولها مهم است**

**\*/**

**SELECT CONCAT(students.firstname , ' ', students.lastname) AS fullname ,**

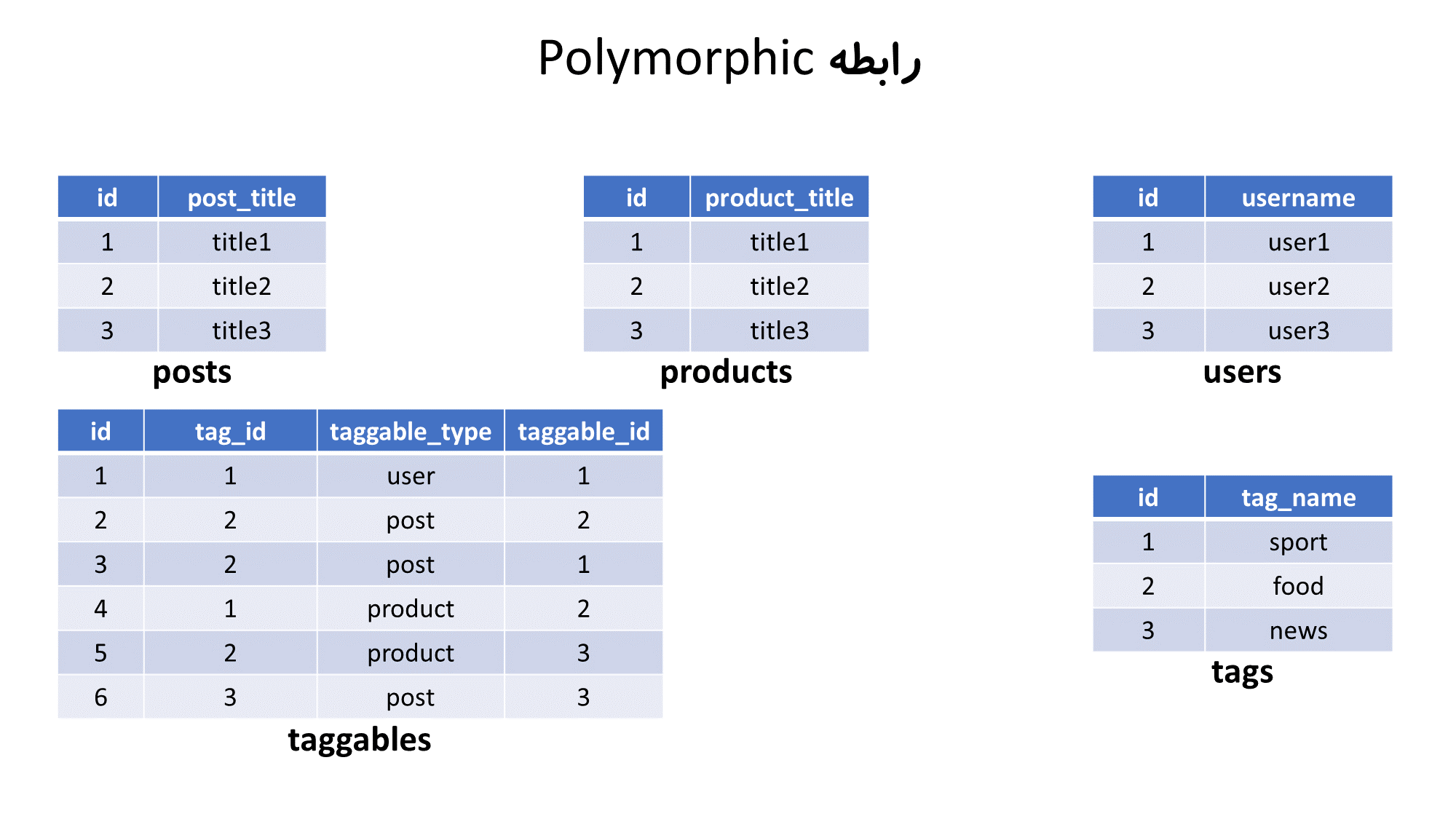
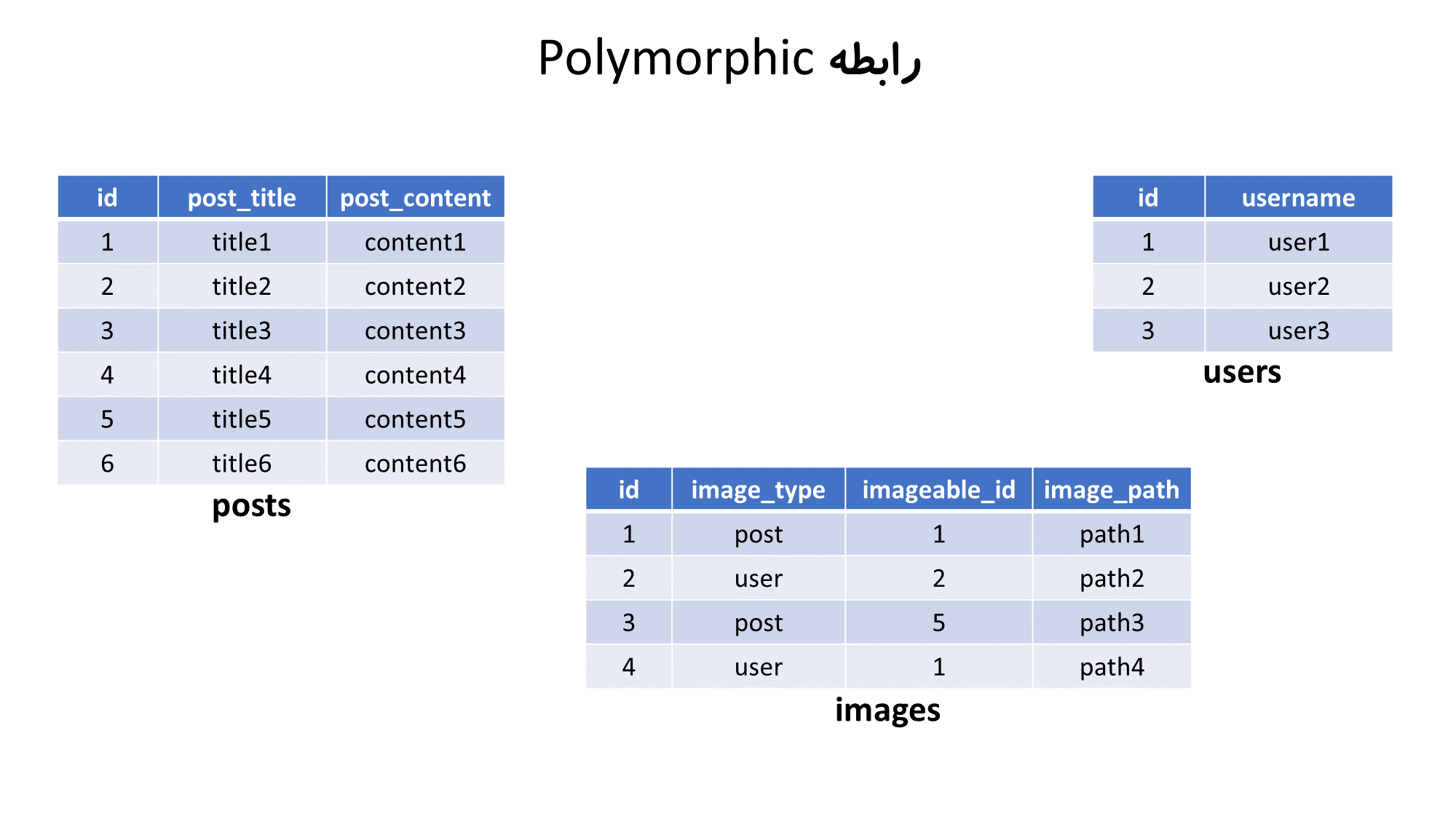
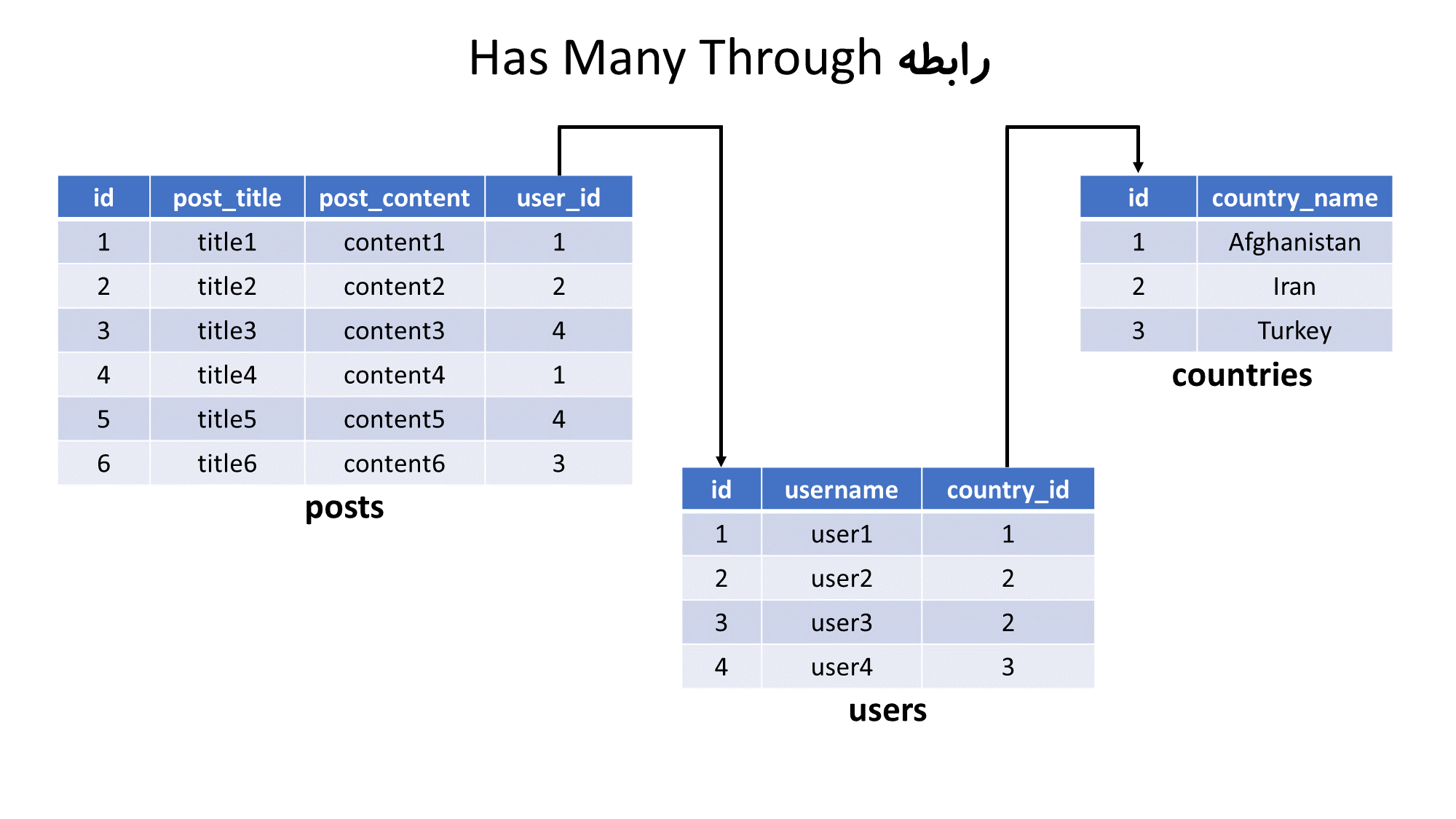
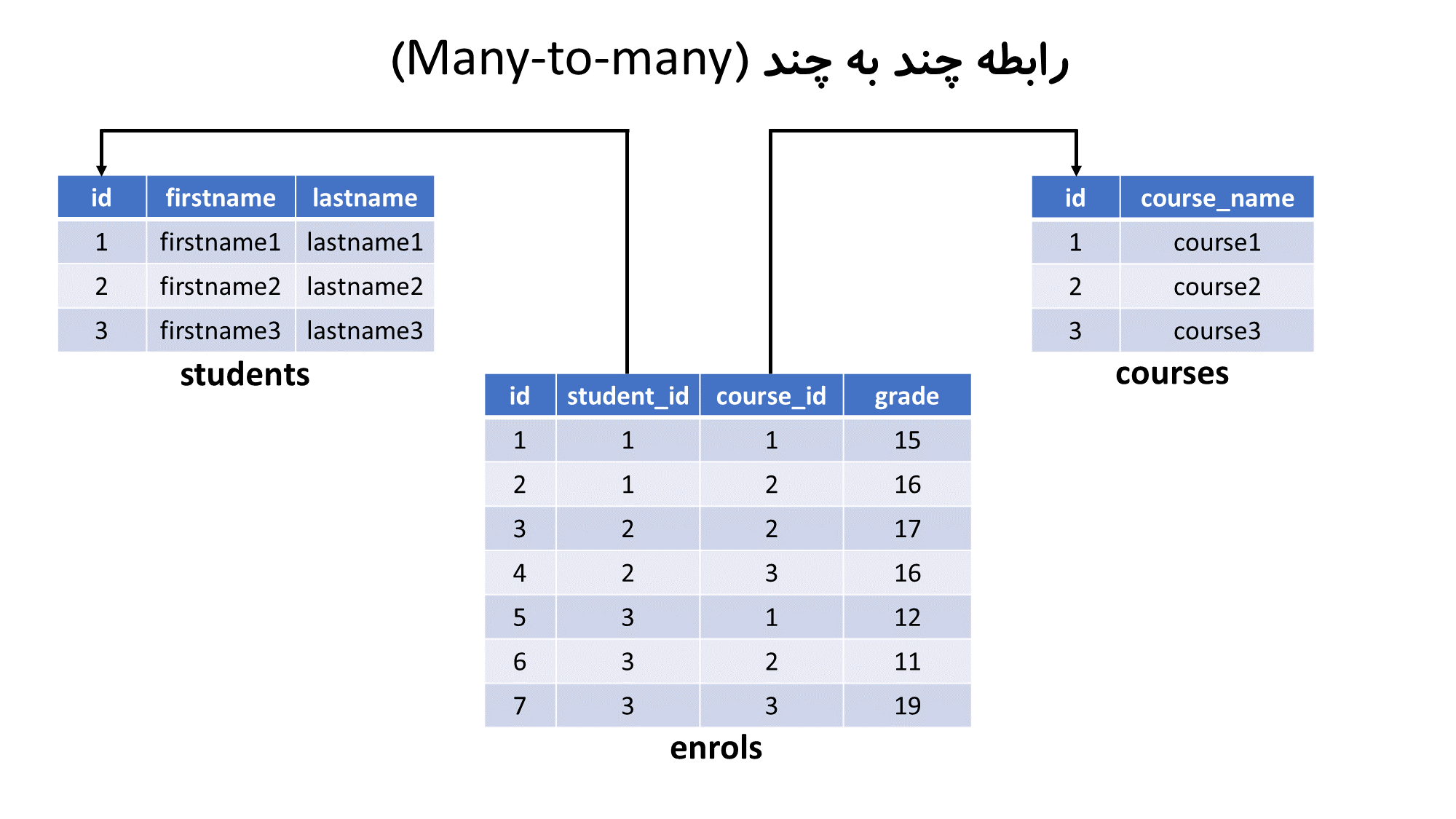
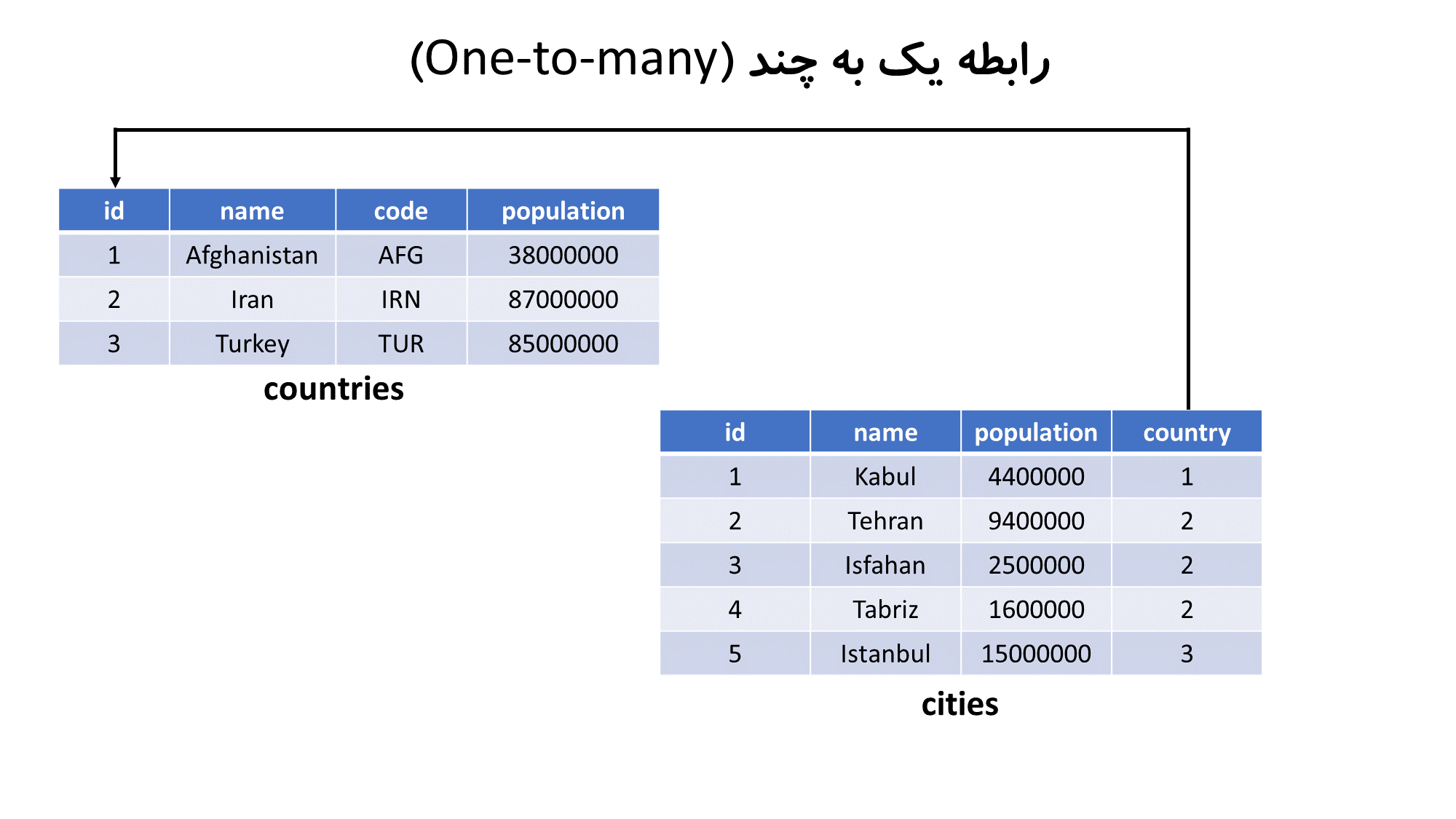
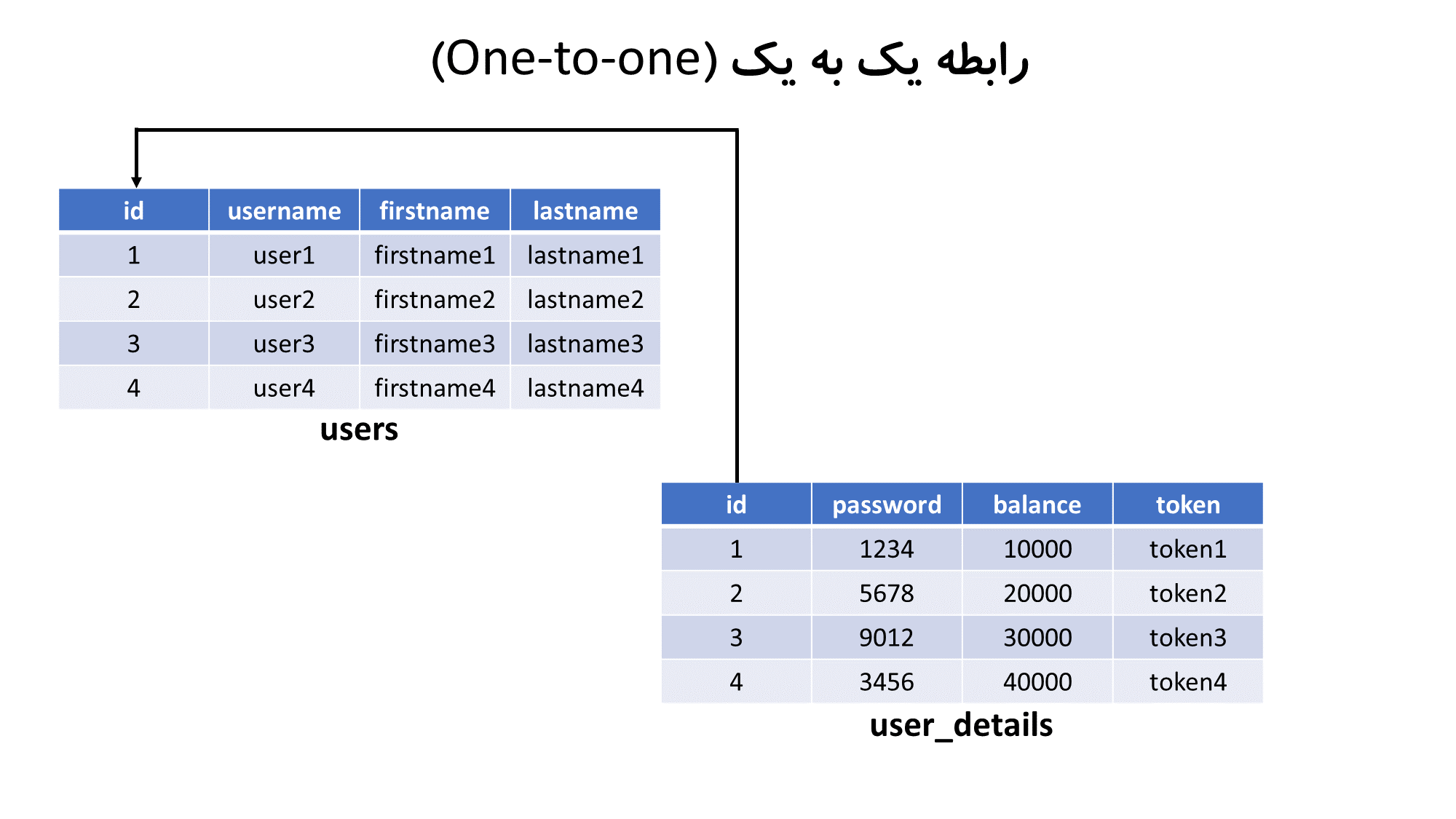
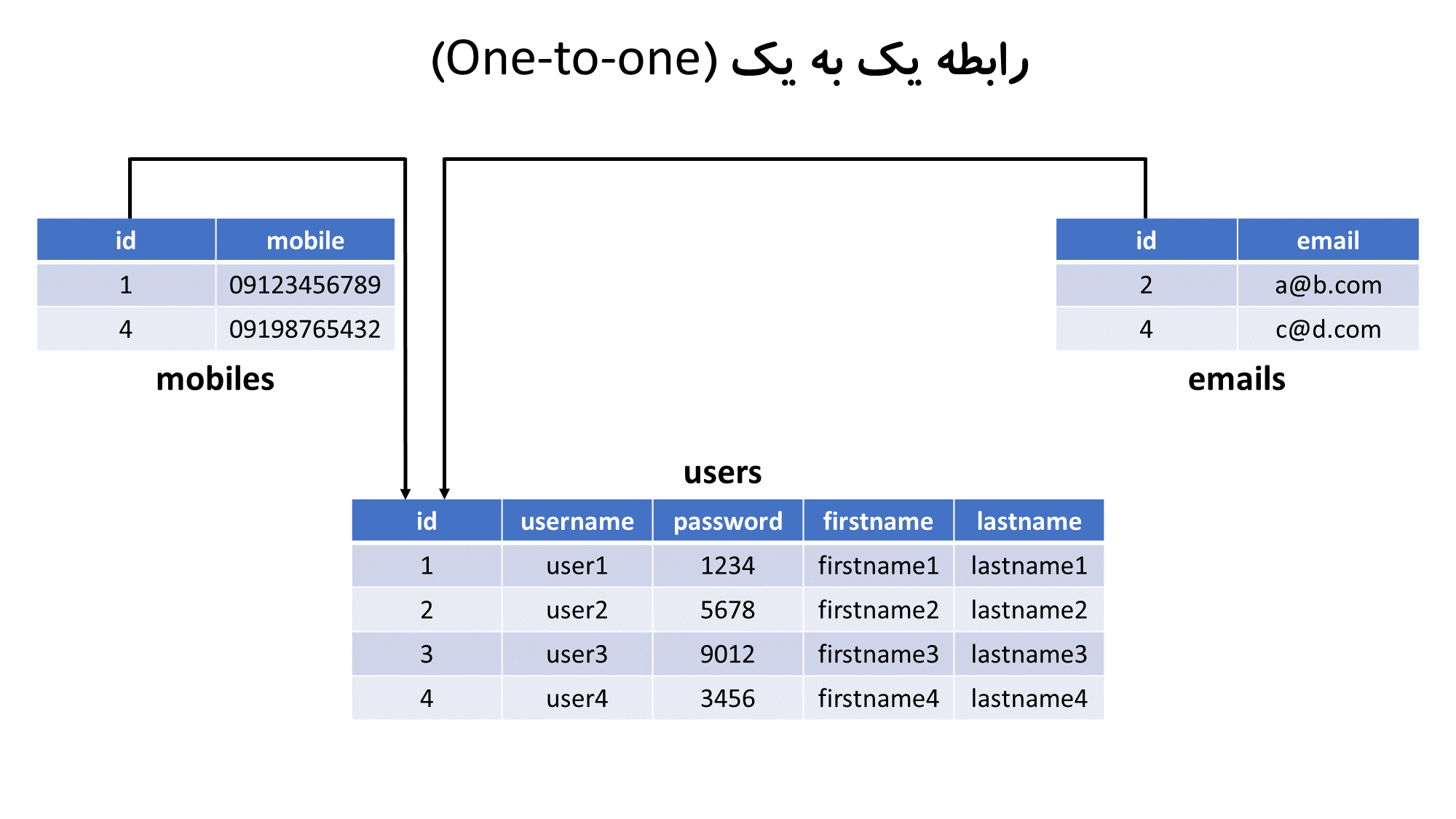
**courses.course\_name , enrols.grade**

**FROM students NATURAL JOIN courses INNER JOIN enrols**

**ON courses.course\_id = enrols.course\_id**

**WHERE grade > 15 ORDER BY grade DESC;**

**27 – انواع روابط بین جدول ها :**

****

**28 – Aggregation Functions**

**SUM – AVG – MIN – MAX – COUNT**

**SELECT \* FROM country;**

**SELECT SUM(Population) , AVG(Population) AS avg\_pop FROM country;**

**SELECT MIN(Population) AS min\_pop , MAX(Population) AS max\_pop FROM country;**

**SELECT COUNT(Population) FROM country;**

**SELECT COUNT(\*) FROM country;**

**SELECT COUNT(Capital) FROM country;**

**/\***

**توابع تجمیعی فیلدهای**

**NULL**

**را در نظر نمی‌گیرند**

**\*/**

**SELECT MIN(Name) , MAX(Name), AVG(Name) , SUM(Name) FROM country;**

**SELECT MIN(Continent) , MAX(Continent), AVG(Continent) , SUM(Continent) FROM country;**

**/\***

**توابع**

**MIN و MAX**

**روی فیلدهای**

**ENUM و SET**

**به صورت رشته ای عمل می‌کنند و توابع**

**AVG و SUM**

**اما به صورت عددی عمل می‌کنند**

**\*/**

**/\***

**توابع**

**MIN و MAX**

**بر روی فیلدهای از نوع تاریخ و زمان هم قابل استفاده هستند**

**\*/**

**29 – Group By**

**SELECT \* FROM country;**

**SELECT \* FROM country GROUP BY Continent; -- Error**

**/\***

**در حالت پیش فرض، در صورت استفاده از**

**GROUP BY**

**مقابل کلمه کلیدی**

**SELECT**

**فقط می توان از فیلدهای به کار رفته در بخش**

**GROUP BY**

**استفاده کرد**

**\*/**

**SET sql\_mode=(SELECT REPLACE(@@sql\_mode,'ONLY\_FULL\_GROUP\_BY',''));**

**SELECT Continent FROM country GROUP BY Continent;**

**SELECT Continent , Region FROM country GROUP BY Continent , Region;**

**SELECT Continent , SUM(Population) AS `sum` , MAX(Population) AS `max` , MIN(Population) AS `min`**

**FROM country GROUP BY Continent;**

**SELECT Code , MAX(Population) FROM country; -- Error**

**/\***

**در حالت پیش فرض، استفاده از توابع تجمعی به همراه سایر فیلدها در مقابل کلمه کلیدی**

**SELECT**

**بدون استفاده از**

**GROUP BY**

**امکان پذیر نیست**

**\*/**

**SELECT CountryCode FROM city GROUP BY CountryCode;**

**SELECT CountryCode FROM city GROUP BY CountryCode ORDER BY Population; -- Error**

**-- فقط فیلدهایی که در گروه بندی شرکت کرده باشند، قابل استفاده برای مرتب سازی هستند**

**SELECT CountryCode FROM city GROUP BY CountryCode ORDER BY CountryCode DESC;**

**SELECT CountryCode , SUM(Population) , MAX(Population) , MIN(Population)**

**FROM city GROUP BY CountryCode;**

**SELECT SUM(Population) FROM city;**

**SELECT SUM(Population) FROM city WHERE Population > 500000;**

**SELECT SUM(Population) , COUNT(Population) FROM city WHERE Population > 500000;**

**SELECT CountryCode , SUM(Population) , MAX(Population) , MIN(Population)**

**FROM city WHERE Population > 500000 GROUP BY CountryCode;**

**SELECT CountryCode , COUNT(\*) AS c FROM city**

**WHERE Population > 500000 GROUP BY CountryCode;**

**SELECT CountryCode , COUNT(\*) AS c FROM city**

**WHERE Population > 500000 AND c > 3 GROUP BY CountryCode; -- Error**

**/\***

**از توابع تجمعی و همچنین نام مستعار آنها نمی‌توان در بخش**

**WHERE**

**استفاده کرد.**

**\*/**

**SELECT CountryCode , COUNT(\*) FROM city**

**WHERE Population > 500000 GROUP BY CountryCode WITH ROLLUP;**

**SELECT CountryCode ,**

**COUNT(\*) , SUM(Population) , MIN(Population) , MAX(Population) , AVG(Population)**

**FROM city WHERE Population > 500000 GROUP BY CountryCode WITH ROLLUP;**

**Group and join :**