Relational Database

1405 Instructor: Ruiqing (Sam) Cao

Design of Relational Database

- A **relational database** is…

  a collection of data tables

  structured to store and organize data

  that adhere to a **relational model**, which defines

    the relationships between the tables

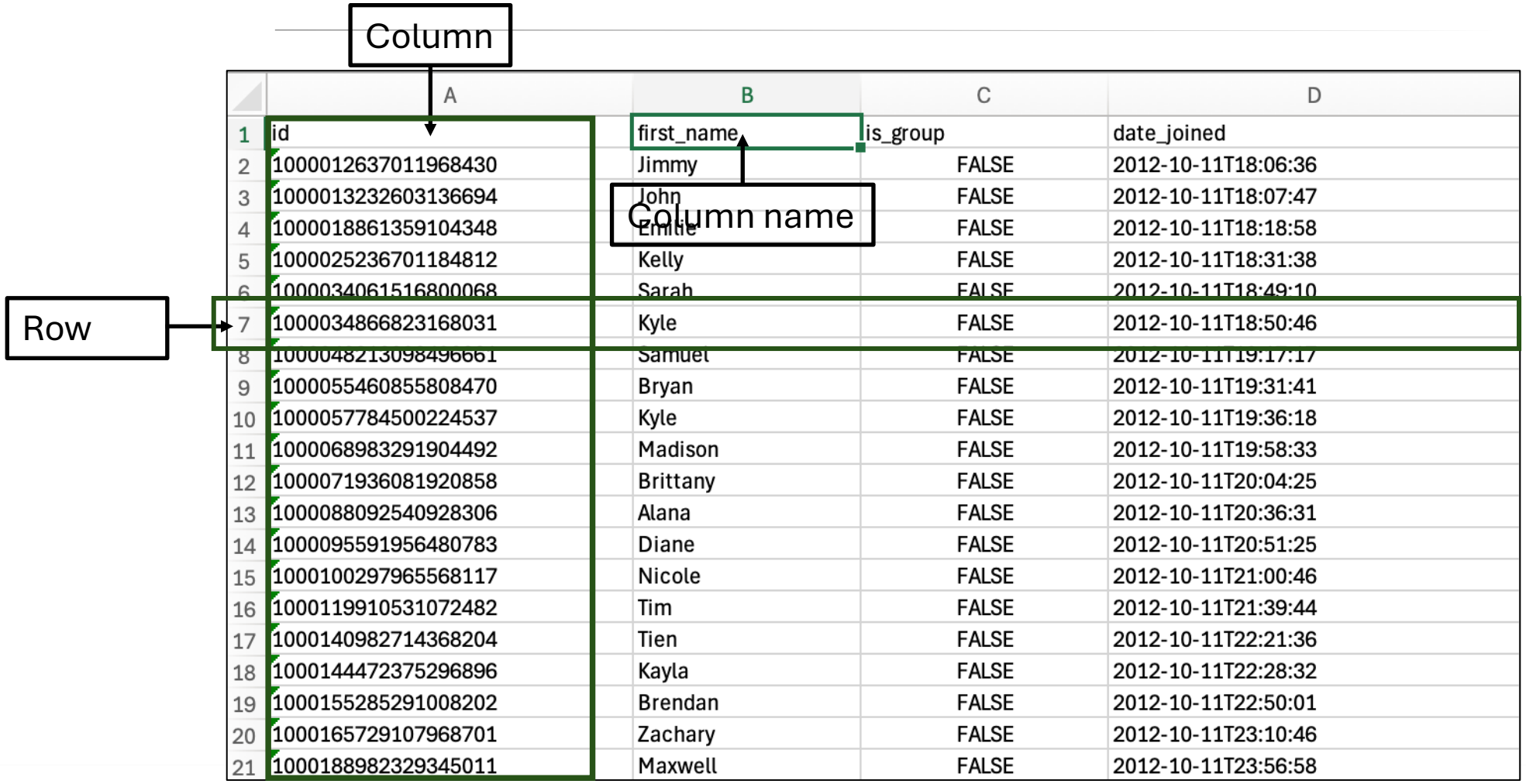
    and the variables contained in them

Relational Model: Table

A **Table** (relation) is a 2D table representing data about an entity

- Each **row** contains data about an instance of the entity, and each **column** represents an attribute of the entity
- Each column has a distinct name, and contains entries of the same data type
- Orders of columns or rows are unimportant

Relational Model: Table



	A	B	C	D
1	id	first_name	is_group	date_joined
2	1000012637011968430	Jimmy	FALSE	2012-10-11T18:06:36
3	1000013232603136694	John	FALSE	2012-10-11T18:07:47
4	1000018861359104348	Emilie	FALSE	2012-10-11T18:18:58
5	1000025236701184812	Kelly	FALSE	2012-10-11T18:31:38
6	1000034061516800068	Sarah	FALSE	2012-10-11T18:49:10
7	1000034866823168031	Kyle	FALSE	2012-10-11T18:50:46
8	1000048213098496661	Samuel	FALSE	2012-10-11T19:17:17
9	1000055460855808470	Bryan	FALSE	2012-10-11T19:31:41
10	1000057784500224537	Kyle	FALSE	2012-10-11T19:36:18
11	1000068983291904492	Madison	FALSE	2012-10-11T19:58:33
12	1000071936081920858	Brittany	FALSE	2012-10-11T20:04:25
13	1000088092540928306	Alana	FALSE	2012-10-11T20:36:31
14	1000095591956480783	Diane	FALSE	2012-10-11T20:51:25
15	1000100297965568117	Nicole	FALSE	2012-10-11T21:00:46
16	1000119910531072482	Tim	FALSE	2012-10-11T21:39:44
17	1000140982714368204	Tien	FALSE	2012-10-11T22:21:36
18	1000144472375296896	Kayla	FALSE	2012-10-11T22:28:32
19	1000155285291008202	Brendan	FALSE	2012-10-11T22:50:01
20	1000165729107968701	Zachary	FALSE	2012-10-11T23:10:46
21	1000188982329345011	Maxwell	FALSE	2012-10-11T23:56:58

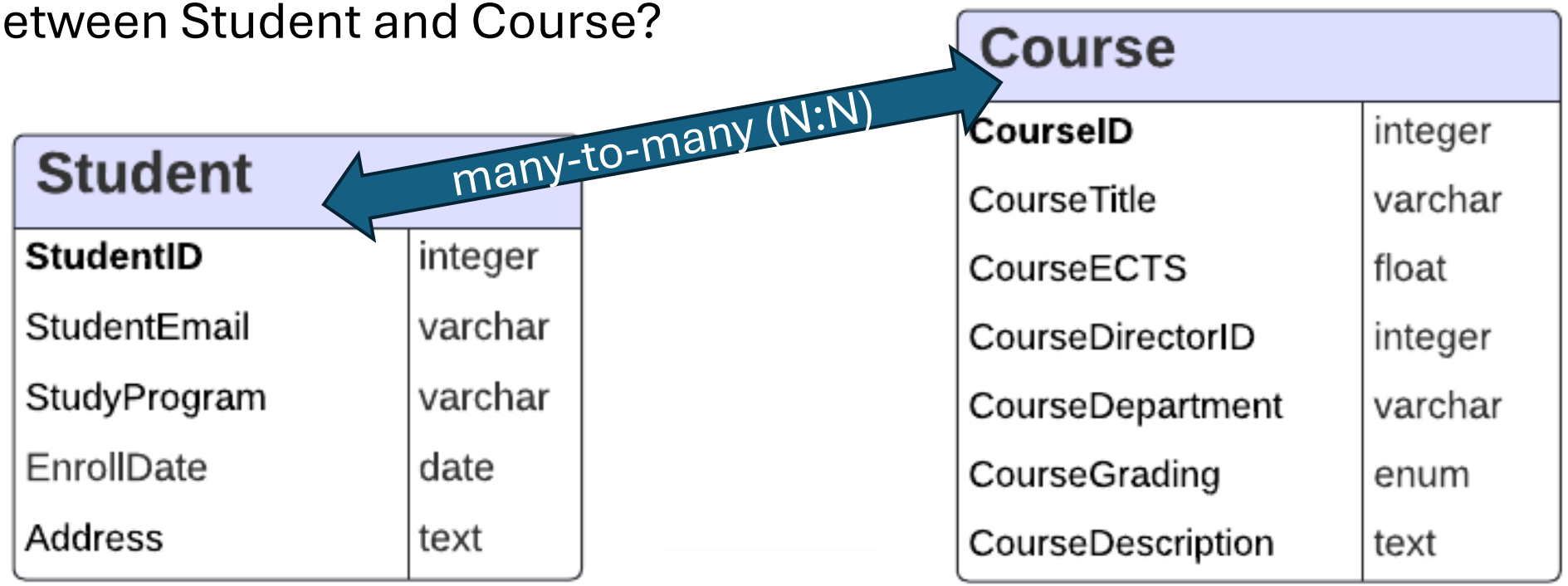
Relational Model: Relationship

Relationships define how two or more data tables are associated with each other, implemented through *primary key* and *foreign key*, and have four types of *cardinalities*:

- One-to-one (1:1)
- One-to-many (1:N)
- Many-to-one (N:1)
- Many-to-many (N:N)

Relational Model: Relationship

- Below are two tables: **Student** and **Course**. A student can sign up for multiple courses (e.g., up to 4), and a course can enroll multiple students (e.g., up to 30).
- What is the **cardinality** of the relationship (in course registration records) between Student and Course?



Relational Model: Primary Key

Candidate key: A minimal set of one or more columns that can uniquely identify each row in a table (can have null values)

Primary Key: One of the candidate keys *selected* to *uniquely identify* each row, and cannot have null values

Example: In a data table representing all students at SSE, the registration number and the email address are both candidate keys, and we can select the registration number as the primary key

Relational Model: Primary Key

- **Primary Key**: One of the candidate keys *selected* to *uniquely identify* each row, and cannot have null values
- **Composite key**: Primary key consisting of two or more attributes
- **Surrogate key**: A unique, numeric value *added* to the table to serve as primary key (no intrinsic meaning, introduced because natural composite keys do not exist or are too complex)

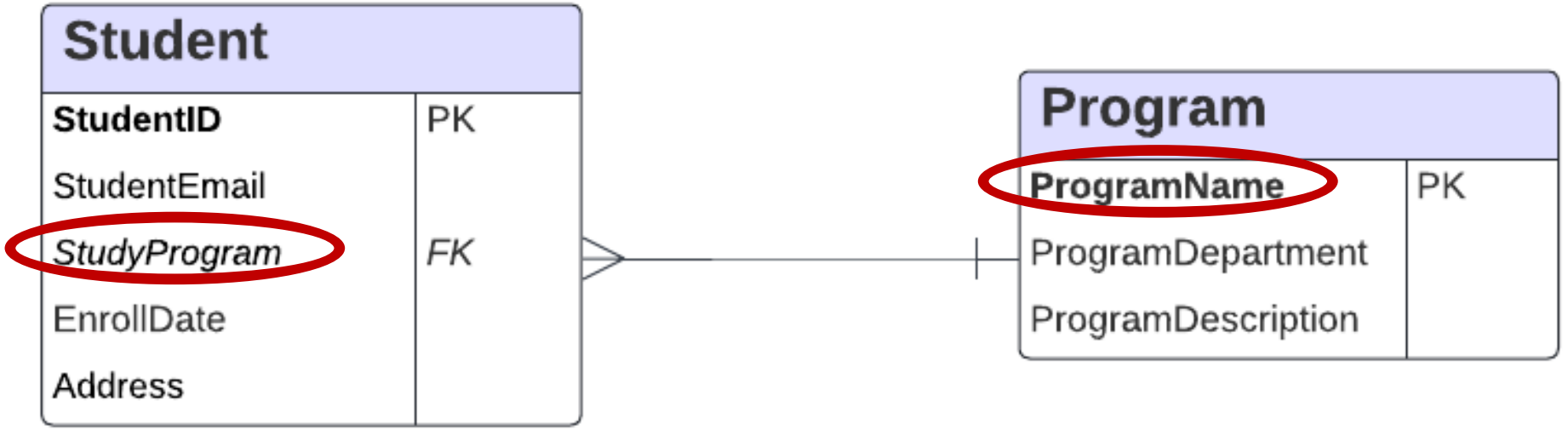
Relational Model: Foreign Key

Foreign Key: a set of columns in a table that establishes a link between two tables.

- The foreign key in one table points to the *primary key in another table*, which **defines a relationship** between the tables
- The foreign key *does not necessarily uniquely identify* each row
- The foreign key *does not necessarily* have the *same name as the primary key* in the other table it points to

Relational Model: Foreign Key

Example: *StudyProgram* is the foreign key in the **Student** table that establishes the link between **Student** and **Program** through linking to the primary key ProgramName in the **Program** table.



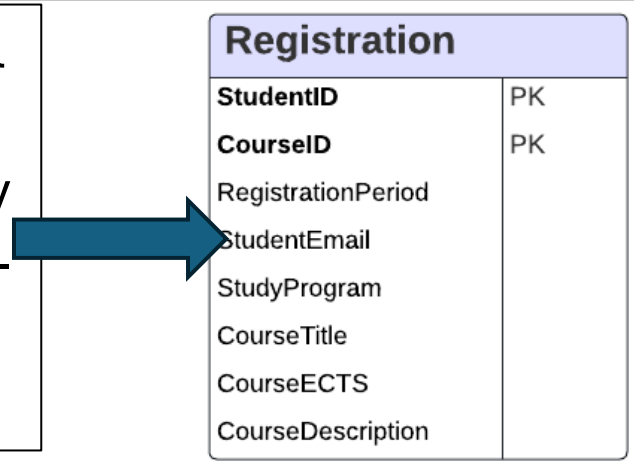
Many-to-Many Relationships

- Many-to-many (N:N) relationships are **not ideal** in database system design

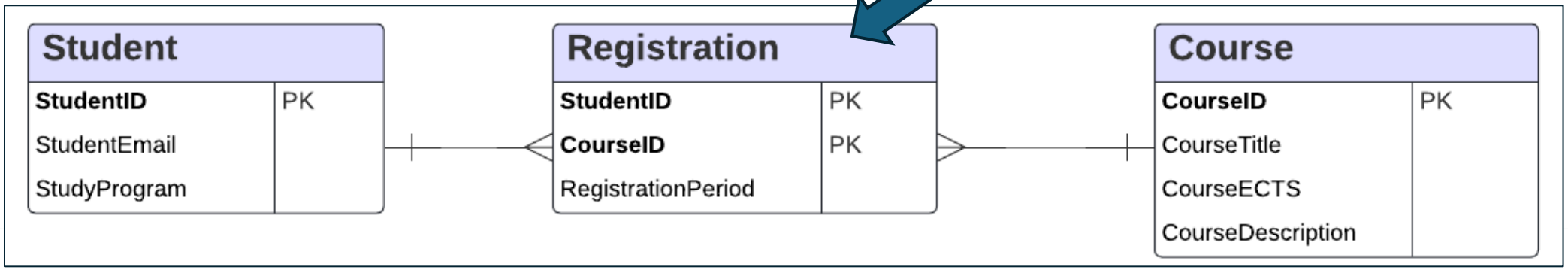
- We need to break tables with many-to-many (N:N) relationships down into smaller chunks and link them back together using primary key and foreign key (part of the normalization process)

Many-to-Many Relationships

Before normalization (partial dependencies of attributes related only to Student or only to Course): contains many-to-many relationship between entities Student and Course



After normalization (no partial dependencies left): split into three tables, with only one-to-many (1:N) and many-to-one (N:1) relationships between tables



Relational Model: Constraints

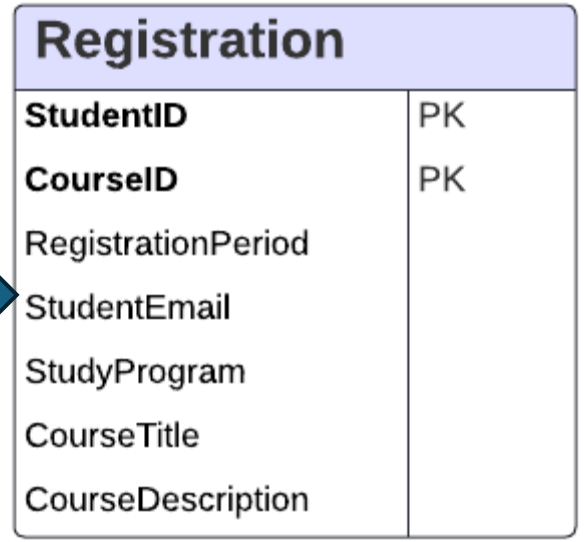
Some examples (not exhaustive of all possibilities):

- *Value* constraint: e.g., income must not be negative
- *Uniqueness* constraint: e.g., a student can only have one school email address
- *Cardinality* constraint: e.g., a student can take at least 1 and no more than 4 classes per period
- *Type* constraint (special case of *domain* constraint): e.g., name must be alphabetical, date must be of the format YYYYMMDD

Anomalies and Data Integrity

- Without *normalization*, modifying a database with new information can result in **anomalies** and cause inconsistencies
- ***Insertion anomaly***: an instance with null values on a variable cannot be inserted into the table

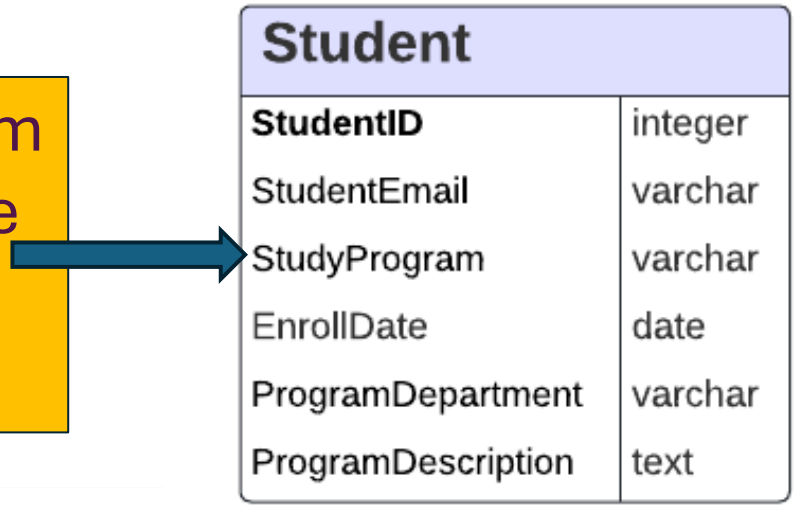
Example: a student enrolled but haven't yet registered for any courses (studentID and StudentEmail are available, but no courseID or any course-related data is entered)



Anomalies and Data Integrity

- Without *normalization*, updating a database with new information can give rise to **anomalies** and lead to inconsistent data
- ***Update anomaly***: same information is stored in multiple rows, and updating one row requires updating all related rows

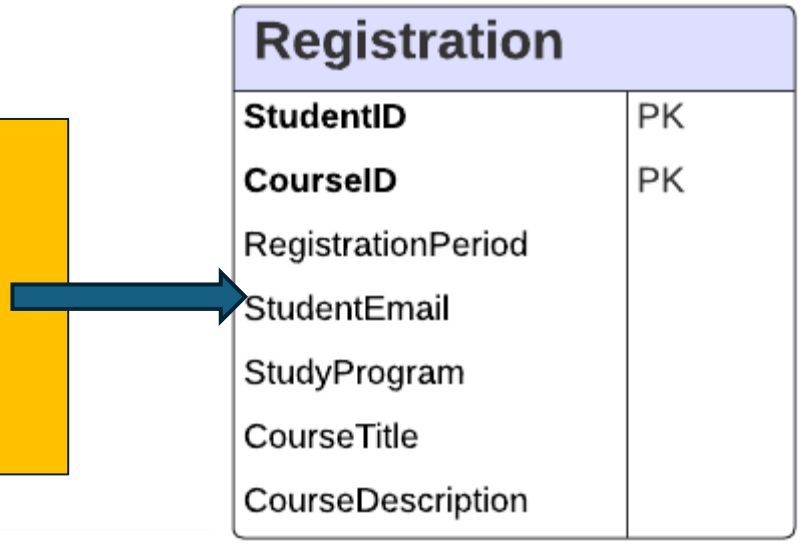
Example: when a change to the study program requires updating ProgramDescription for the MBM program, it needs to be updated for all students enrolled in the MBM program



Anomalies and Data Integrity

- Without *normalization*, updating a database with new information can give rise to **anomalies** and lead to inconsistent data
- ***Deletion anomaly***: deleting a piece of unwanted data causes desired information to be deleted as well

Example: when the last student enrolled in the course 1405 drops it, no record of the class will remain in the data even though the class itself still exists



Database Normalization

- **Database normalization** is an approach developed over the years to address issues of *various anomalies*

- Normalization is the process of **breaking down a large table into smaller tables** that adhere to certain rules, and **linking them together with foreign keys**

- Failing to normalize data can lead to several types of anomalies that *compromise data integrity* and *lead to inconsistent data*

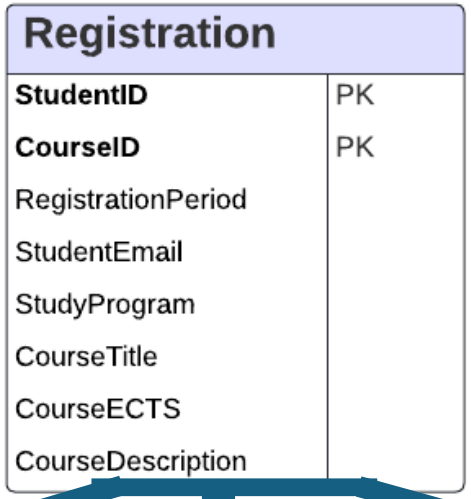
Normalization and Normal Forms

- **Normalization** solve the anomalies by reconstructing the database to remove redundancies in the data

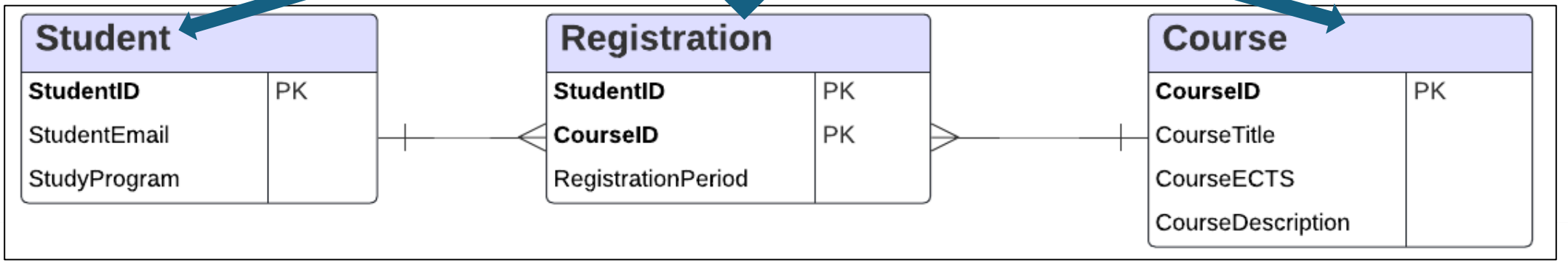
- A series of normal forms sequentially (1st, 2nd, 3rd are the most common) defined by the previous one and some additional rules
  - ***1st Normal Form (1NF)*** ensures every column contains the same type of data with atomic, indivisible values, and there is no repeating groups or arrays, and each row is unique
  - ***2nd Normal Form (2NF)*** requires 1NF and all non-prime attributes (not part of a candidate key) fully functionally dependent on the primary key
  - ***3rd Normal Forms (3NF)*** requires 2NF and all non-prime attributes depend only on the primary key (not on other non-prime attribute)

Normalization and Normal Forms

Before normalization:



After normalization:



Normal Forms: Example

- Read the (hypothetical) student registration records table ("registration_records_anon.csv") into Jupyter Notebook

- We'll demonstrate normal forms using this table as an example

	program	student_id	status	registered	registered_year	registered_month	registered_day	student_phone	student_country	student_city
0	MFIN Corporate Fin. & Investment Mgmt	32	Student	28/08/2024	2024	8	28	0704******	Sverige	Stockholm
1	BSc in Business and Economics 2020	32	Degree completed 2024	23/08/2021	2021	8	23	0704******	Sverige	Stockholm
2	MFIN Corporate Fin. & Investment Mgmt	45	Student	28/08/2024	2024	8	28	 0709******		
3	BSc in Business and Economics 2020	45	Degree completed 2024	23/08/2021	2021	8	23	 0709******		
4	Exchange: MSc in Business & Economics	34	Student	20/01/2025	2025	1	20	NaN	NaN	NaN

1st Normal Form (1NF)

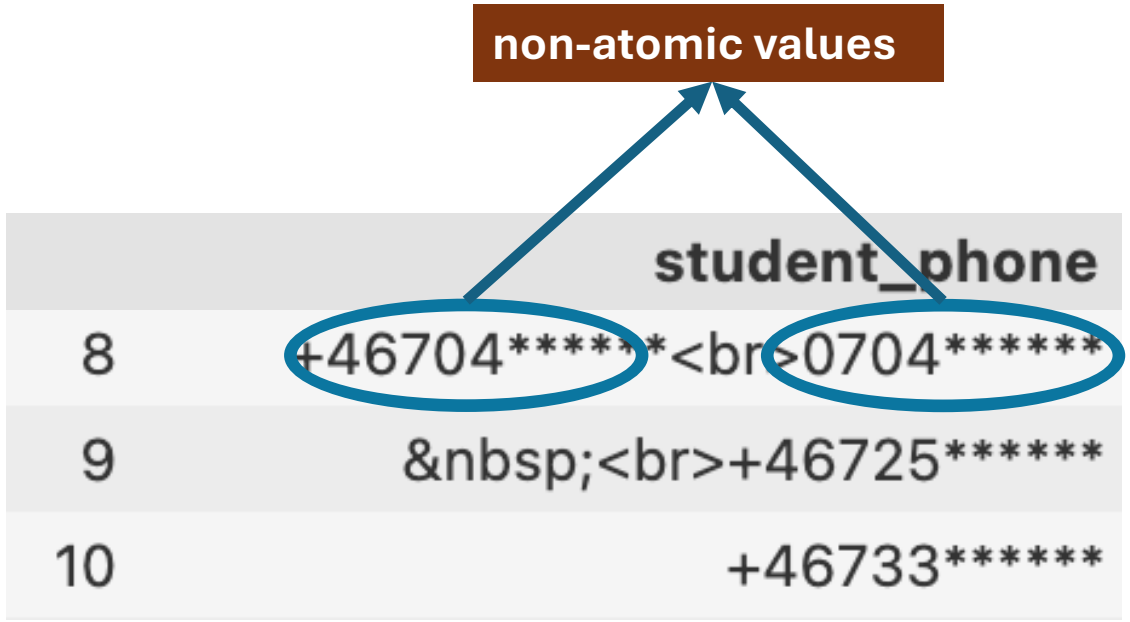
- ***1NF*** ensures every column contains the **same type of data** with **atomic, indivisible values**, and there is **no repeating groups** or arrays, and **each row is unique**

For example:

- A column cannot store two different types of values (e.g., cannot have studentName and departmentName in the same column in different rows)
- A column or columns cannot store more than one value (e.g., cannot have *a list of* phone numbers in the same column, or *multiple columns* for several phone numbers)

1st Normal Form (1NF): Example

- In the example of student registration records table (see Notebook for details): `student_phone` contains non-atomic values (multiple phone numbers). **This violates 1NF.**



2nd Normal Form (2NF)

- ***2NF*** requires 1NF and all non-prime attributes (not part of a candidate key) *fully functionally dependent* on the primary key
  - **Fully functional dependency**: a column is determined by the ***full*** primary key; partial dependency violates 2NF (a column is determined by only part (and not all of the composite primary key)
- 2NF is fundamentally about ***organizing tables into separate "themes"*** (e.g., Student, Program, etc) by ensuring that *all non-prime attributes are determined by ALL parts of the primary key*

2nd Normal Form (2NF): Example

- In this example, `student_country` (also `student_phone` and `student_city`) only depends on `student_id`.
- `student_country` is only partially (not fully) dependent on the entire primary key (`program`,`student_id`)
- **This violates 2NF.**

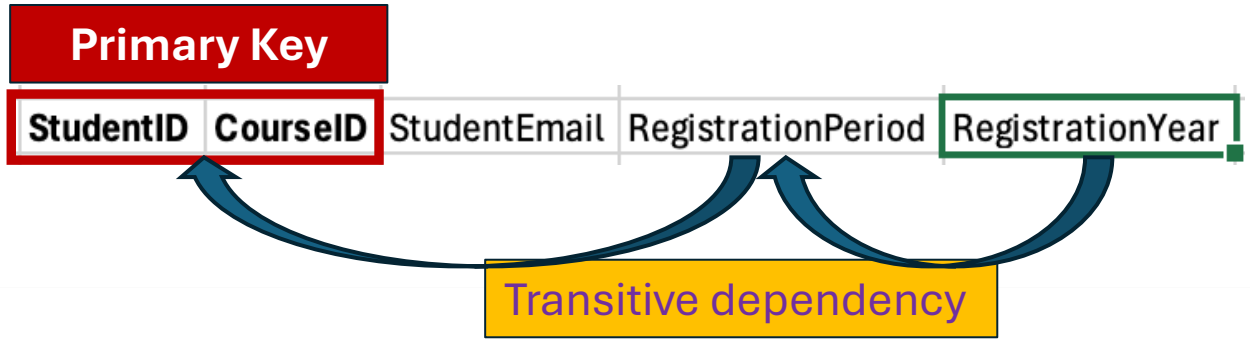
partial dependency

	program	student_id	student_country	student_phone	student_city
0	MFIN Corporate Fin. & Investment Mgmt	32	Sverige	0704******	Stockholm
1	BSc in Business and Economics 2020	32	Sverige	0704******	Stockholm
2	MFIN Corporate Fin. & Investment Mgmt	45		 0709******	
3	BSc in Business and Economics 2020	45		 0709******	
4	Exchange: MSc in Business & Economics	34	NaN	NaN	NaN
5	MAVFM Accounting, Valuation & Financial Mgmt	31	Sverige	+46708******	Johanneshov
6	BSc in Business and Economics 2020	31	Sverige	+46708******	Johanneshov

3rd Normal Form (3NF)

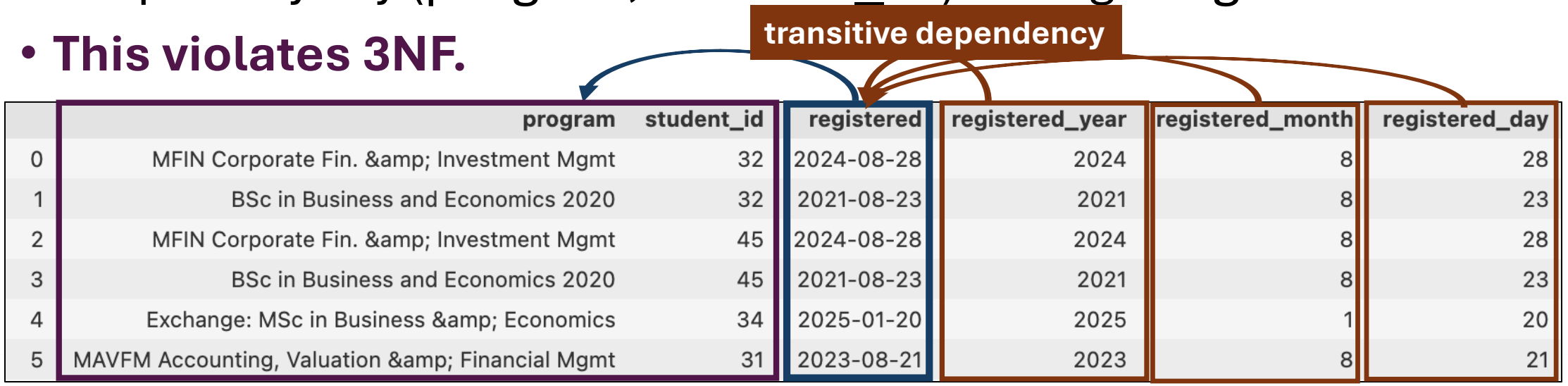
- ***3NF*** requires 2NF and all non-prime attributes *depend only on the primary key* (not on other non-prime attribute)
  - **Eliminate transitive dependency**: a column depends on a non-prime attribute (which in turn depends on the primary key)

- 3NF removes indirect dependency between non-prime attributes and the primary key through intermediate attributes



3rd Normal Form (3NF): Example

- In this example, `registered_year` (also `registered_month` and `registered_day`) depends entirely on `registered` (a non-prime attribute, i.e., not part of a candidate key).
- `registered_year` is only indirectly (transitively) dependent on the primary key (`program`, `student_id`) through `registered`.
- **This violates 3NF.**



	program	student_id	registered	registered_year	registered_month	registered_day
0	MFIN Corporate Fin. & Investment Mgmt	32	2024-08-28	2024	8	28
1	BSc in Business and Economics 2020	32	2021-08-23	2021	8	23
2	MFIN Corporate Fin. & Investment Mgmt	45	2024-08-28	2024	8	28
3	BSc in Business and Economics 2020	45	2021-08-23	2021	8	23
4	Exchange: MSc in Business & Economics	34	2025-01-20	2025	1	20
5	MAVFM Accounting, Valuation & Financial Mgmt	31	2023-08-21	2023	8	21

Denormalization and Unnormalized Form

- **Sometimes we do not want full normalization** (e.g., analytical databases and dimensional model where data structure is better kept simple)

➢ **Denormalization**: recombine some tables back together after they had been split apart to conform to normalization rules

- **Unnormalized Form (UNF)**: when raw data is stored without any formal structure or normalization applied (e.g., unstructured data collected from real-world sources, one big CSV file with all data)

Data Model Design & Data Operations

- Translating a data model into a practical relational database with a structured schema requires dealing with the reality of messy data which is often the case with real-world data sources

- **Basic data operations** useful for data cleaning:
  - **Filtering**: selecting a *subset of rows* satisfying on certain *conditions*
  - **Projection**: selecting a *subset of columns*
  - **Union**: *combining rows* and removing duplicates
  - **Join**: *combine columns* based on common data elements

Summary of Data Modeling Key Points

- Database design needs to take into consideration different ***entities*** (such as transactions, products, consumers) and the ***relationships*** between them
- Changes are constantly applied to the database, which will require modifying the data in more than one place, thus the design needs to ensure ***data integrity*** throughout the entire system (by applying the ***data normalization*** process)
- ***Transactional*** and ***analytical*** data have distinct features and require different modeling approaches to balance data integrity with structural simplicity (***ER Model*** vs. ***Dimensional Model***)

Exercise: Data Normalization

The dataset **crunchbase_europe_2016_2020.csv** contains information about funding rounds and organizational profiles of European startups founded between 2016 and 2020. Your task is to evaluate the data structure from a database design perspective and improve it by applying normalization techniques.

1. Examine the CSV File: Review the data and understand its structure. Look at the columns and rows to identify key entities, relationships, and any redundancy.
2. Identify Potential Problems: Discuss the issues in the current structure, such as update anomalies, deletion anomalies, and insertion anomalies
3. Does the current structure violate 1NF? Write a modified structure that satisfies 1NF
4. Does the modified structure violate 2NF? Write a modified structure that satisfies 2NF
5. Does the modified structure violate 3NF? Write a modified structure that satisfies 3NF