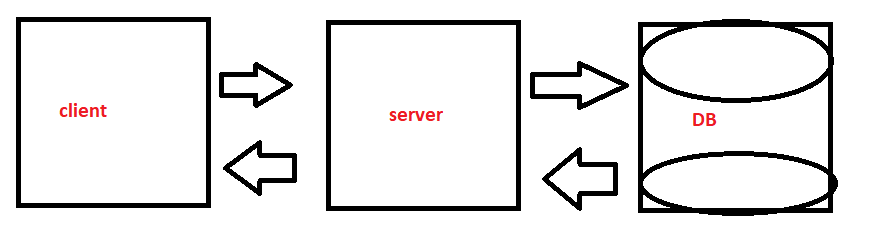
**Database**

**מסד נתונים**

מערכת המאפשרת לשמור ולנהל כמות גדולה של נתונים.

המערכת הינה חיצונית לאפליקציה שלנו ; היא תמיד נמצאת ב backend, האפליקציה שלנו ניגשת אליה ומשתמשת בה.



**קיימים שני סוגים מרכזיים של DB:**

1. מסד נתונים רלציוני – relational database

זה בעצם מסד נתונים השומר את המידע שלו בתוך טבלאות; למשל:

Mysql, postgress, MSSQL, Oracle, MariaDB

1. מסד נתונים NoSQL

זה הוא כל מסד נתונים השומר את המידע שלו בצורה אחרת. אין כאן טבלאות, לרוב זה Json like – דוגמאות :Firestore, elastic, Firebase, cassandara וMongoDB

הם הרבה יותר מהירים במסדי נתונים רלציונים, ולכן משתמשים בהם ב big data

**Relational data base**

**Table**

טבלה. המידע נמצא בטבלאות ; טבלה מכילה מידע מסוג מסוים, אין לערבב מספר סוגי נתונים באותה טבלה.

לדוגמה : טבלה של לקוחות – לא תחזיק מידע על מוצרים.

**Column**

עמודה. מאפיין מוגדר בעמודה.

**Row/Record**

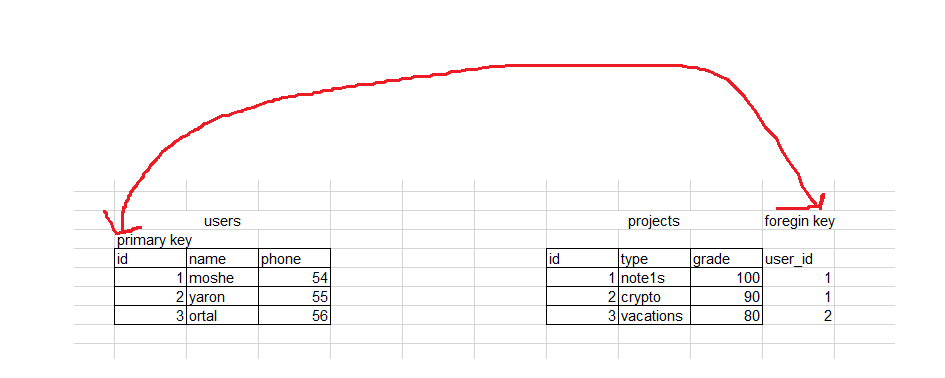
שורה – המידע עצמו.

**Primary key**

זו עמודה המכילה מזהה ייחודי עבור כל שורה.

Foreign key

זו עמודה המכילה ערכי primary key של טבלה אחרת – בכדי לדעת מי בטבלה האחרת שייך לה.



כמו בתכנות, גם על מסדי נתונים – נהוג להסתכל כעל ישויות ומאפיינים ( entities and attributes ).

המידע שלנו ( ה data ) מחולק לישויות, כאשר חלק מהן (לא כולן) קשורות אחת לשניה ( relationship)

ישויות – entities , זה בעצם ה " מה" ; "על מה המידע שלנו"

מאפיינים – attributes - זה בעצם התיאור של אותן ישויות.

|  |  |
| --- | --- |
| Attributes | Entities |
| * The student name * The student address * The classes they are taken | * College – can store information about: * The student * The professor * The classes * The majors |
| Database – college  Tables (entities) – students, professors, classes  Rows/Records(attributes) name, address, classes etc… | |

**Relation**

קשר בין שתי ישויות ( טבלאות) או יותר.

* למשל טבלת סטודנטים, מקושרת איכשהו לטבלת קורסים (סטודנט הולך לאוניברסיטה ולוקח קורסים).
* למשל טבלת מקצועות, מקושרת איכשהו גם היא לטבלת המרצים ( הרי כל מרצה, מלמד קורס מסוים).

**One to One Relationship**

**One entity has a connection with one another entity**

זה הוא מצב שבו, ליישות מסוימת קיים קשר יחיד ( דו כיווני) עם ישות אחת אחרת.

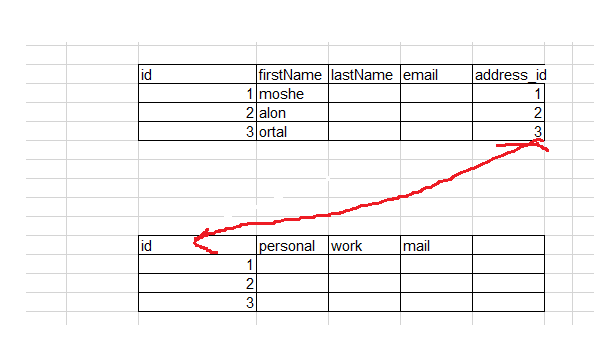
**למשל:** לקוח ומספר ת.ז

כדי לתאר one to one , אנחנו יכולים לומר שלכל לקוח , יש מספר ת.ז אחד בלבד; כאשר מספר ת.ז אחד בלבד יכול להיות מקשור ללקוח אחד בלבד.

Database design

* לרוב, אנחנו בעצם לא מתייחסים לישות השני
* ה, כישות עצמאית. אלא כאל מאפיין ( attribute) . כלומר, נוצר מצב שבו במקום ששם המשתמש ות.ז של המתמש, והאימייל של המשתמש, ישמשו כישויות נפרדות , הן נוצרות (ישויות) כמאפיינים באותה טבלה.
* זה כמובן תלוי בכמות המידע שיש לאותו מאפיין. במידה ואנו רואים שיש כמות מידע גדולה, אנחנו יכולים לנהל את זה בשתי טבלאות. למשל – משתמשים וכתובות ; מצד אחד, לכל משתמש יכולה להיות כתובת מגורים אחת;

אבל יכול להיות שהאפליקציה שלנו, צריכה גם כתובת למשלוח דואר, כתובת נוספת שהיא לא כתובת מגורים, כתובת של מקום העבודה, כתובת של ההורים; ואז, במצב כזה אולי שווה לעשות טבלה אחת של כתובות.

****

**One to Many relationships**

**One entity can have a relationship with multiple other entities – but a specific entity can only be related back to one entity**

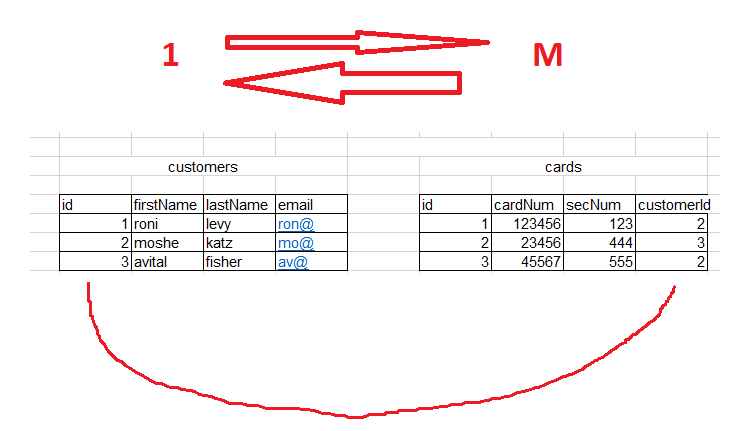
זה הוא מצב שבו לישות אחת, יכול להיות קשר עם מספר ישויות, אבל לכיוון אחד בלבד.

למשל : טבלת לקוחות וטבלת כרטיסי אשראי.

אם תהיה לנו טבלה שבה נשמור את הלקוחות שלנו וטבלת נוספת שבה נשמור את מספרי כרטיס האשראי שלהם, כדי לתאר One to many, אנחנו יכולים לומר כך: לכל לקוח, יכולים להיות כמה כרטיסי אשראי ; אבל כרטיס אשראי יכול להיות שייך ללקוח אחד בלבד.

Database design

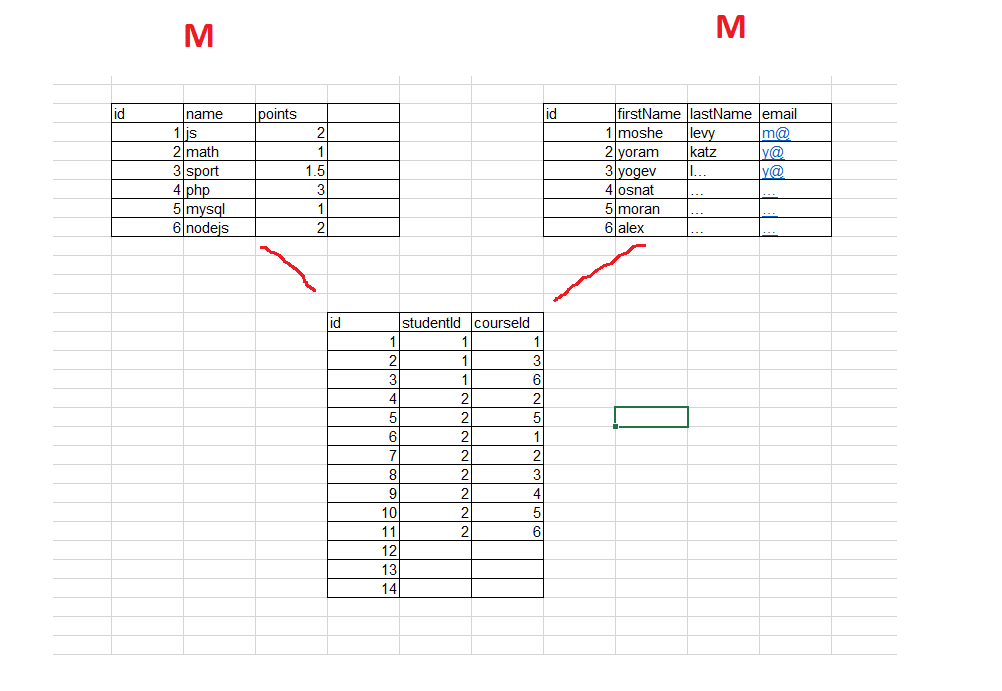
* אנחנו מפרידים את הישויות לטבלאות נפרדות;
* כאשר הטבלה של ה many, מקבלת כ reference את ה Primary key של הטבלה של ה ONE ומשתמשת בו כ foreign key
* כלומר, לטבלת ה customers, יהיה שדה של primary key (ID) והוא יהיה מוזן כ foreign key בטבלה של כרטיסי האשראי



Many to many relationship

Many entities can have a relationship with multiple other entities

* זה הוא מצב שבו לישות אחת, יכול להיות קשר עם מספר ישויות, אבל זה דו כיווני.
* למשל : קורסים וסטודנטים
* אם תהיה לנו טבלה של סטודנטים, וטבלה של קורסים, אז כדי לתאר את ה Many to many, אחננו יכולים לומר שכל סטודנט יכול לקחת מספר קורסים , אבל כל קורס, יכול להילקח גם על ידי מספר סטודנטים.
* במקרה של Many to many, אנחנו נתקלים בסוג של בעיה. אנחנו לא יכולים לשמור foreign key, כי יש כמה בכל צד. לכן, אנחנו הופכים כל ישות כזו ל One to many, ונעזרים במשהו שנקרא טבלת נרמול , שהיא תייצג את ה many to many



**סוגי טבלאות**

טבלת בסיס – Base Table

זו טבלה המכילה מידע ראשי/ מרכזי במערכת שלנו.

לדוגמה : טבלת תלמידים, מרצים, חומרי עזר, מוצרים, ספרים, סופרים – וכו. תמיד ניתן להוסיף, לעדכן, למחק מידע כזה מהמערכת.

טבלת פירוט – detailing table

זו טבלה המכילה פירוט/ מידע לגבי רשומות בטבלת בסיס.

לדוגמה – ציונים עבור תלמידים, טלפונים עבור מרצים , מלאי מוצרים.

תמיד טבלת פירוט תהיה קשור לטבלת הבסיס שלה עם קשר one to many, המוגדר כך שכאשר נמחק רשומה בטבלת הבסיס, כל הרשומות הקשורות אליה מטבלת הפירוט יימחקו גם הן (פיצ'ר זה נקרא cascading).

Intermediate table

זו היא טבלה המייצגת קשר של רבים לרבים בין שתי טבלאות; כי לא ניתן לבצע קשר כזה בצורה ישירה. לכן הטבלה הזו, מייצגת את הקשר הזה; בינה, יש קשר לכל טבלה של יחיד לרבים ( One to many)