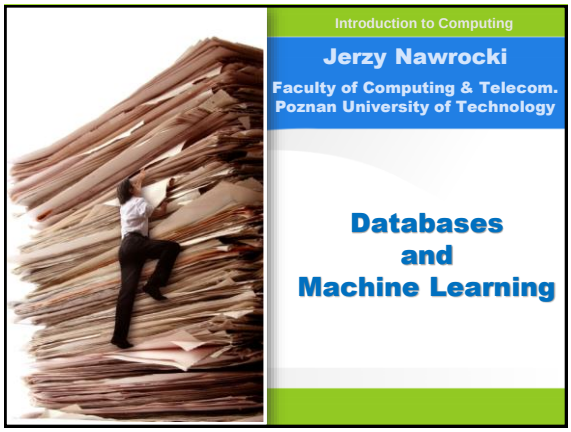
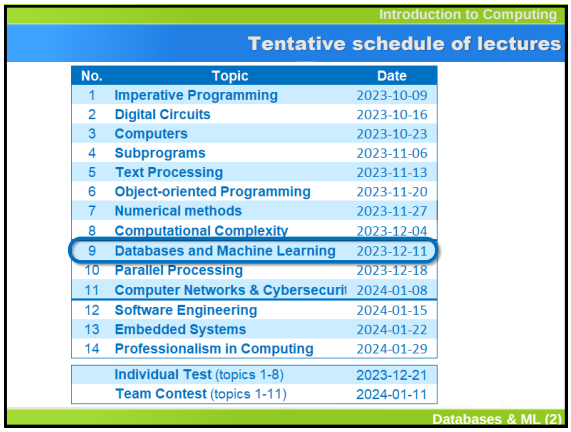
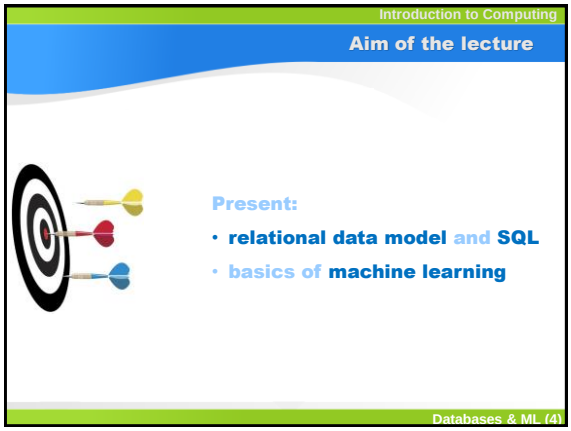
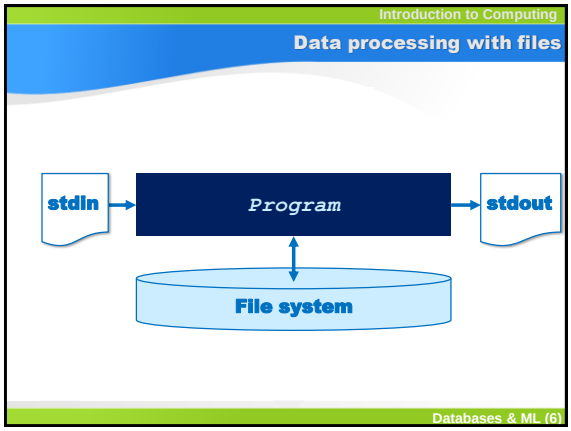

Introduction to Computing
Jerzy Nawrocki
Faculty of Computing & Telecom.
Poznan University of Technology
Databases
and
Machine Learning


Introduction to Computing
Tentative schedule of lectures
No. Topic Date
1 Imperative Programming 2023-10-09
2 Digital Circuits 2023-10-16
3 Computers 2023-10-23
4 Subprograms 2023-11-06
5 Text Processing 2023-11-13
6 Object-oriented Programming 2023-11-20
7 Numerical methods 2023-11-27
8 Computational Complexity 2023-12-04
9 Databases and Machine Learning 2023-12-11
10 Parallel Processing 2023-12-18
11 Computer Networks & Cybersecurit 2024-01-08
12 Software Engineering 2024-01-15
13 Embedded Systems 2024-01-22
14 Professionalism in Computing 2024-01-29
Individual Test (topics 1-8) 2023-12-21
Team Contest (topics 1-11) 2024-01-11
Databases & ML (2)


Introduction to Computing
Database
= organized collection of data
Databases & ML (3)

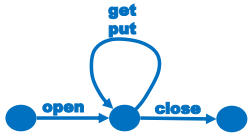

Introduction to Computing
Aim of the lecture
Present:
• relational data model and SQL
• basics of machine learning
Databases & ML (4)


Introduction to Computing
Agenda
• File processing
• Relational databases
• Basics of SQL
• Advanced features of SQL
• Python interface to SQLite3
• Machine learning & ID3
Databases & ML (5)


Introduction to Computing
Data processing with files
stdin
Program
stdout
File system
Databases & ML (6)

Introduction to Computing

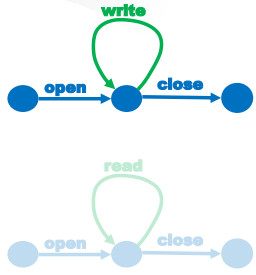
Drawer metaphor



https://www.ikea.com/ca/en/p/malm-3-drawer-chest-black-brown-30473149/

Introduction to Computing

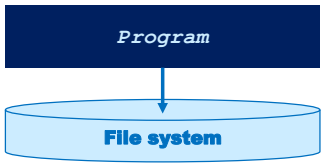
Outline of file processing



https://www.ikea.com/ca/en/p/malm-3-drawer-chest-black-brown-30473149/

Introduction to Computing

Writing to a file



Introduction to Computing

Writing to a file

File name

Mode (w = write)

```
file= open("Data.txt", "w")
file.write("Hello!")
file.close()
```

Data.txt

```
Hello!
```


Introduction to Computing

Writing to a file

File name

Mode (w = write)

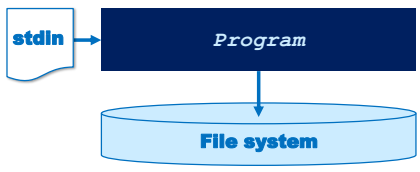
```
file= open("Data.txt", "w")
file.write("Hello!")
file.close()
```

C

```
#include <stdio.h>
int main() {
  FILE *fptr;
  fptr = fopen("Data.txt", "w");
  fprintf(fptr, "Hello!");
  fclose(fptr);
  return 0; }
```


Introduction to Computing

Schema of data processing



Writing to a file

stdin

```
Sunny    Hot  High    Strong No
Overcast Hot  High    Weak   Yes
```

```
import sys
file= open("Data.txt", "w")
file.write("Outlook  Temp Humidity Wind   Opinion\n")
for line in sys.stdin:
    file.write(line)
file.close()
```

Data.txt

```
Outlook  Temp Humidity Wind   Opinion
Sunny    Hot  High     Strong No
Overcast Hot  High     Weak   Yes
```


Writing to a file

```
import sys
file= open("Data.txt", "w")
file.write("Outlook  Temp Humidity Wind   Opinion\n")
for line in sys.stdin:
    file.write(line)
file.close()
```

```
#include <stdio.h>
int main() {
  char line[99];
  FILE *fptr;
  fptr = fopen("Data.txt", "w");
  fprintf(fptr,
          "Outlook  Temp Humidity Wind   Opinion\n");
  while (fgets(line, 99, stdin) != NULL){
        fprintf(fptr, "%s", line); }
  fclose(fptr);
  return 0; }
```


Writing to a file

```
import sys
file= open("Data.txt", "w")
file.write("Outlook  Temp Humidity Wind   Opinion\n")
for line in sys.stdin:
    file.write(line)
```

```
#include <stdio.h>
int main() {
  char line[99];
  FILE *fptr;
  fptr = fopen("Data.txt", "w");
  fprintf(fptr,
          "Outlook  Temp Humidity Wind   Opinion\n");
  while (fgets(line, 99, stdin) != NULL){
        fprintf(fptr, "%s", line); }

  return 0; }
```

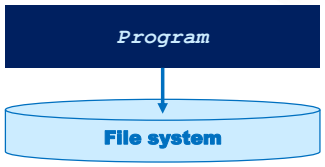

Writing to a file

```
import sys
file= open("Data.txt", "w")
file.write("Outlook  Temp Humidity Wind   Opinion\n")
for line in sys.stdin:
    file.write(line)
```

```
#include <stdio.h>
int main() {
  char line[99];
  FILE *fptr;
  fptr = fopen("Data.txt", "w");
  fprintf(fptr,
          "Outlook  Temp Humidity Wind   Opinion\n");
  while (fgets(line, 99, stdin) != NULL){
        fprintf(fptr, "%s", line); }

  return 0; }
```


Appending to a file



Data.txt

```
Outlook  Temp Humidity Wind   Opinion
Sunny    Hot  High     Strong No
Overcast Hot  High     Weak   Yes
```


Appending to a file

Mode (a = append)

```
file= open("Data.txt", "a")
file.write("\nRain     Mild High     Strong No")
file.close()
```

Data.txt

```
Outlook  Temp Humidity Wind   Opinion
Sunny    Hot  High     Strong No
Overcast Hot  High     Weak   Yes
Rain     Mild High     Strong No
```

Introduction to Computing

Appending to a file

Mode (a = append)

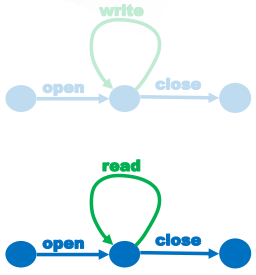
```
file= open("Data.txt", "a")
file.write("\nRain      Mild High      Strong No")
file.close()
```

C

```
#include <stdio.h>
int main() {
  FILE *fptr;
  fptr = fopen("Data.txt", "a");
  fprintf(fptr,
          "\nRain      Mild High      Strong No");
  fclose(fptr);
  return 0; }
```


Introduction to Computing

Outline of file processing



https://www.ikea.com/ca/en/p/malm-3-drawer-chest-black-brown-30473149/

Introduction to Computing

Schema of data processing

Data.txt

Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No

Introduction to Computing

Appending to a file

Mode (r = read)

```
values= []
file= open("Data.txt", "r")
for line in file:
    v= line.split()[1]
    if values.count(v) == 0:
        values.append(v)
file.close()
for v in values:
    print(v+' ', end="")
```

```
Hot Mild Cool
```

Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No

C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char line[99];
    char v0[22], v1[22], v2[22], v3[22], v4[22];
    char values[5][22];
    int  top= 0, count, i;
    FILE *fptr;
    fptr = fopen("Data.txt", "r");
    while (fscanf(fptr, "%s %s %s %s %s",
                  v0, v1, v2, v3, v4) != EOF){
        count= 0;
        for (i= 0; i < top; i++)
            if (strcmp(values[i], v1) == 0)
                count++;
        if (count == 0)
           strcpy(values[top++], v1); }
    fclose(fptr);
    printf("%s", values[0]);
    for (i= 1; i < top; i++)
        printf(" %s", values[i]);
    return 0; }
```

C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char line[99];
    char v0[22], v1[22], v2[22], v3[22], v4[22];
    char values[5][22];
    int  top= 0, count, i;
    FILE *fptr;
    fptr = fopen("Data.txt", "r");
    while (fscanf(fptr, "%s %s %s %s %s",
                  v0, v1, v2, v3, v4) != EOF){
        count= 0;
        for (i= 0; i < top; i++)
            if (strcmp(values[i], v1) == 0)
                count++;
        if (count == 0)
           strcpy(values[top++], v1); }
    fclose(fptr);
    printf("%s", values[0]);
    for (i= 1; i < top; i++)
        printf(" %s", values[i]);
    return 0; }
```

C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char line[99];
    char v0[22], v1[22], v2[22], v3[22], v4[22];
    char values[5][22];
    int  top= 0, count, i;
    FILE *fptr;
    fptr = fopen("Data.txt", "r");
    while (fscanf(fptr, "%s %s %s %s %s",
                  v0, v1, v2, v3, v4) != EOF){
        count= 0;
        for (i= 0; i < top; i++)
            if (strcmp(values[i], v1) == 0)
                count++;
        if (count == 0)
            strcpy(values[top++], v1); }
    fclose(fptr);
    printf("%s", values[0]);
    for (i= 1; i < top; i++)
        printf(" %s", values[i]);
    return 0; }
```

C

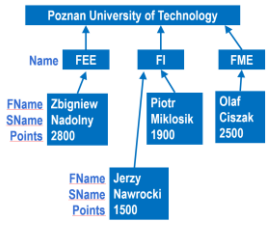
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char line[99];
    char v0[22], v1[22], v2[22], v3[22], v4[22];
    char values[5][22];
    int  top= 0, count, i;
    FILE *fptr;
    fptr = fopen("Data.txt", "r");
    while (fscanf(fptr, "%s %s %s %s %s",
                  v0, v1, v2, v3, v4) != EOF){
        count= 0;
        for (i= 0; i < top; i++)
            if (strcmp(values[i], v1) == 0)
                count++;
        if (count == 0)
            strcpy(values[top++], v1); }
    fclose(fptr);
    printf("%s", values[0]);
    for (i= 1; i < top; i++)
        printf(" %s", values[i]);
    return 0; }
```


Agenda

- File processing
- **Relational databases**
- Basics of SQL
- Advanced features of SQL
- Python interface to SQLite3
- Machine learning & ID3

Hierarchical database model

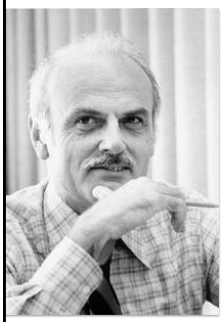
Hierarchical database model



„In order to retrieve data from a hierarchical database the whole tree needs to be traversed starting from the root node."

https://en.wikipedia.org/wiki/Hierarchical_database_model

Edgar Frank Codd



1923-08-23: Born in Portland, Dorset
Studied math and chemistry at Oxford
RAF pilot during II World War
1948: IBM, New York
1963: Ph.D., Univ. of Michigan, Ann Arbor
1970: *„A Relational Model of Data for Large Shared Data Banks"*
1981: Turing Award
2003-04-18: Died in Williams Island, Florida (US)

Formal approach

Workers

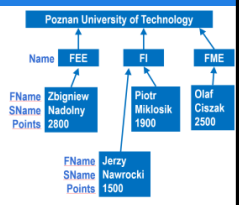
FName	SName	Points
Olaf	Ciszak	2 500
Piotr	Miklosik	1 900
Jerzy	Nawrocki	1 500
Zbigniew	Nadolny	2 800

Workers: FName x Sname x Points
FName = {"a",.., "z", "aa",.., "zz",.., "A",.., "Z",.., "Aa",.., "Az", ..}
SName = {"a",.., "z", "aa",.., "zz",.., "A",.., "Z",.., "Aa",.., "Az", ..}
Points = 0, 1, 2, ...
Workers= { ("Olaf", "Ciszak", 2 500), ("Piotr", "Miklosik", 1 900), .., ("Zbigniew", "Nadolny", 2 800) }

Tables can be extended

Employment

Worker	Faculty
20	1
21	2
22	2
23	3



Faculties

Key	Name
1	FME
2	FI
3	FEE

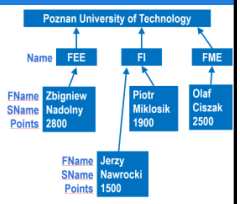
Workers

Key	FName	SName	Points
20	Olaf	Ciszak	2 500
21	Piotr	Miklosik	1 900
22	Jerzy	Nawrocki	1 500
23	Zbigniew	Nadolny	2 800

Tables can be extended

Employment

Worker	Faculty
20	1
21	2
22	2
23	3



Faculties

Key	Name	Dean
1	FME	20
2	FI	22
3	FEE	23

Workers

Key	FName	SName	Points
20	Olaf	Ciszak	2 500
21	Piotr	Miklosik	1 900
22	Jerzy	Nawrocki	1 500
23	Zbigniew	Nadolny	2 800

Agenda

- File processing
- Relational databases
- **Basics of SQL**
- Advanced features of SQL
- Python interface to SQLite3
- Machine learning & ID3

SQL

Structured Query Language

Designed by:
Donald D. Chamberlin &
Raymond F. Boyce

Appeared in 1974

ANSI standard in 1986
ISO standard in 1987

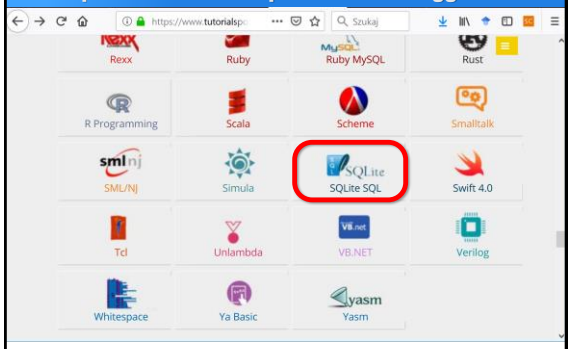
Recommended for experimentation

SQLite

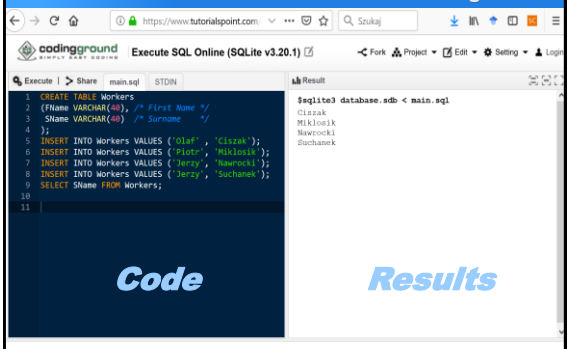
Designed in 2000 by D.Richard **Hipp**
Single file database (up to 2 TB)
+ Relatively easy to combine with Python
+ Very fast (2-3 times faster than MySQL and PostgreSQL for many common operations)
- For writing the whole file gets locked

D. Richard Hipp

https://www.tutorialspoint.com/codingground.htm



SQLite at Coding Ground



An operational model of data

C R U D

CREATE **READ** **UPDATE** **DELETE**

CRUD vs. SQL operations

C R U D

CREATE **SELECT** **UPDATE** **DELETE**
INSERT

SQL – Basic operations

CREATE an empty table with a given set of columns
INSERT a row into a given table
SELECT (read) data from table(s)
UPDATE some rows of a given table
DELETE some rows of a given table

CREATE TABLE

```
1  CREATE TABLE Workers
2  (FName VARCHAR(40), /* First Name */
3   SName VARCHAR(40)  /* Surname    */
4  );
```

Workers

FName	SName

Basic data types:
VARCHAR (n)
CHAR (n)
INT
NUMERIC (n.f)

SQL – Basic operations

CREATE an empty table with a given set of columns
INSERT a row into a given table
SELECT (read) data from table(s)
UPDATE some rows of a given table
DELETE some rows of a given table

INSERT

```
1  CREATE TABLE Workers
2  (FName VARCHAR(40), /* First Name */
3   SName VARCHAR(40)  /* Surname    */
4  );
5  INSERT INTO Workers VALUES ('Olaf' , 'Ciszak');
6  INSERT INTO Workers VALUES ('Piotr', 'Miklosik');
7  INSERT INTO Workers VALUES ('Jerzy', 'Nawrocki');
8  INSERT INTO Workers VALUES ('Jerzy', 'Suchanek');
```

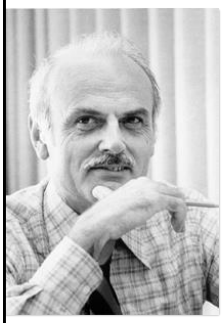
Workers

FName	SName
Olaf	Ciszak
Piotr	Miklosik
Jerzy	Nawrocki
Jerzy	Suchanek

SQL – Basic operations

CREATE an empty table with a given set of columns
INSERT a row into a given table
SELECT (read) data from table(s)
UPDATE some rows of a given table
DELETE some rows of a given table

Edgar Frank Codd



Basic variants of SELECT:

- **projection,**
- **selection,**
- **join**

SELECT – Projection

```
1  CREATE TABLE Workers
2  (FName VARCHAR(40), /* First Name */
3   SName VARCHAR(40)  /* Surname    */
4  );
5  INSERT INTO Workers VALUES ('Olaf' , 'Ciszak');
6  INSERT INTO Workers VALUES ('Piotr', 'Miklosik');
7  INSERT INTO Workers VALUES ('Jerzy', 'Nawrocki');
8  INSERT INTO Workers VALUES ('Jerzy', 'Suchanek');
9  SELECT SName FROM Workers;
```

What will the result be?

SELECT – Projection

```
1  CREATE TABLE Workers
2  (FName VARCHAR(40), /* First Name */
3   SName VARCHAR(40)  /* Surname    */
4  );
5  INSERT INTO Workers VALUES ('Olaf' , 'Ciszak');
6  INSERT INTO Workers VALUES ('Piotr', 'Miklosik');
7  INSERT INTO Workers VALUES ('Jerzy', 'Nawrocki');
8  INSERT INTO Workers VALUES ('Jerzy', 'Suchanek');
9  SELECT SName FROM Workers;
```

Workers

FName	SName
Olaf	Ciszak
Piotr	Miklosik
Jerzy	Nawrocki
Jerzy	Suchanek

```
Ciszak
Miklosik
Nawrocki
Suchanek
```

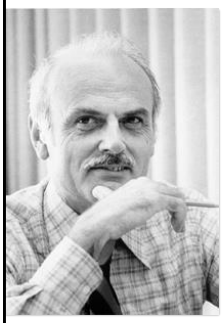
Projection in AWK

```
FName   SName
Olaf    Ciszak
Piotr   Miklosik
Jerzy   Nawrocki
Jerzy   Suchanek
```

```
NR > 1    {print $2;}
```

```
Ciszak
Miklosik
Nawrocki
Suchanek
```


Edgar Frank Codd



Basic variants of SELECT:

- **projection,**
- **selection,**
- **join**

Selection of rows

```
1  CREATE TABLE Workers
2  ( FName VARCHAR(40), /* First Name */
3    SName VARCHAR(40)  /* Surname    */
4  );
5  INSERT INTO Workers VALUES ('Olaf' , 'Ciszak');
6  INSERT INTO Workers VALUES ('Piotr', 'Miklosik');
7  INSERT INTO Workers VALUES ('Jerzy', 'Nawrocki');
8  INSERT INTO Workers VALUES ('Jerzy', 'Suchanek');
9  SELECT * FROM Workers WHERE FName='Jerzy';
```

```
Jerzy|Nawrocki
Jerzy|Suchanek
```


SQLite: the mode command

```
1  .mode column
2  CREATE TABLE Workers
3  ( FName VARCHAR(40), /* First Name */
4    SName VARCHAR(40)  /* Surname    */
5  );
6  INSERT INTO Workers VALUES ('Olaf' , 'Ciszak');
7  INSERT INTO Workers VALUES ('Piotr', 'Miklosik');
8  INSERT INTO Workers VALUES ('Jerzy', 'Nawrocki');
9  INSERT INTO Workers VALUES ('Jerzy', 'Suchanek');
10 SELECT * FROM Workers WHERE FName='Jerzy';
```

```
Jerzy       Nawrocki
Jerzy       Suchanek
```


Selection in AWK

```
FName   SName
Olaf    Ciszak
Piotr   Miklosik
Jerzy   Nawrocki
Jerzy   Suchanek
```

```
$1 == "Jerzy"
```

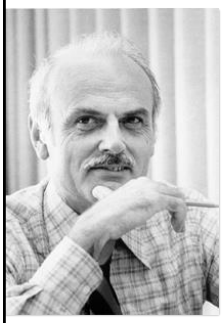
```
Jerzy   Nawrocki
Jerzy   Suchanek
```


SQLite – the header command

```
1  .header on
2  .mode column
3  CREATE TABLE Workers
4  ( FName VARCHAR(40), /* First Name */
5    SName VARCHAR(40)  /* Surname    */
6  );
7  INSERT INTO Workers VALUES ('Olaf' , 'Ciszak');
8  INSERT INTO Workers VALUES ('Piotr', 'Miklosik');
9  INSERT INTO Workers VALUES ('Jerzy', 'Nawrocki');
10 INSERT INTO Workers VALUES ('Jerzy', 'Suchanek');
11 SELECT * FROM Workers WHERE FName='Jerzy';
```

```
FName       SName
----------  ----------
Jerzy       Nawrocki
Jerzy       Suchanek
```

Edgar Frank Codd



Basic variants of SELECT:

- **projection,**
- **selection,**
- **join**

Workers

WFaculty	WFName	WSName	Ext
Me	Olaf	Ciszak	2360
Me	Andrzej	Milecki	2370
Ph	Ryszard	Czajka	3200
Ph	Ryszard	Skwarek	3210
Co	Ryszard	Adamiak	3410
Co	Andrzej	Jaszkiewic	3420
Co	Andrzej	Marciniak	3430
Co	Andrzej	Urbaniak	3440

Deans

DFaculty	DFName	DSName
Co	Andrzej	Jaszkiewicz
Me	Olaf	Ciszak
Ph	Ryszard	Czajka

What is Dean's extension?

```
1  .header on
2  .mode   column
3  CREATE TABLE Workers
4  ( WFaculty VARCHAR(10),  /* Worker's faculty      */
5    /* Co = Computing, Me = Mechanics, Ph = Physics */
6    WFName   VARCHAR(40), /* Worker's first Name    */
7    WSName   VARCHAR(40), /* Worker's surname       */
8    Ext      INT       ); /* Phone number extension */
9  INSERT INTO Workers VALUES ('Me', 'Olaf'   , 'Ciszak',      2360);
10 INSERT INTO Workers VALUES ('Me', 'Andrzej', 'Milecki',     2370);
11 INSERT INTO Workers VALUES ('Ph', 'Ryszard', 'Czajka',      3200);
12 INSERT INTO Workers VALUES ('Ph', 'Ryszard', 'Skwarek',     3210);
13 INSERT INTO Workers VALUES ('Co', 'Ryszard', 'Adamiak',     3410);
14 INSERT INTO Workers VALUES ('Co', 'Andrzej', 'Jaszkiewicz', 3420);
15 INSERT INTO Workers VALUES ('Co', 'Andrzej', 'Marciniak',   3430);
16 INSERT INTO Workers VALUES ('Co', 'Andrzej', 'Urbaniak',    3440);
17 CREATE TABLE Deans
18 ( DFaculty VARCHAR(10),  /* Dean's faculty    */
19   DFName   VARCHAR(40),  /* Dean's first Name */
20   DSName   VARCHAR(40)); /* Dean's surname    */
21 INSERT INTO Deans VALUES ('Co', 'Andrzej', 'Jaszkiewicz');
22 INSERT INTO Deans VALUES ('Me', 'Olaf'   , 'Ciszak'     );
23 INSERT INTO Deans VALUES ('Ph', 'Ryszard', 'Czajka'     );
```


Result

```
SELECT WFaculty, WFName, WSName, Ext
  FROM Workers;
```

WFaculty	WFName	WSName	Ext
Me	Olaf	Ciszak	2360
Me	Andrzej	Milecki	2370
Ph	Ryszard	Czajka	3200
Ph	Ryszard	Skwarek	3210
Co	Ryszard	Adamiak	3410
Co	Andrzej	Jaszkiewic	3420
Co	Andrzej	Marciniak	3430
Co	Andrzej	Urbaniak	3440

A little modification

```
SELECT WFaculty, WFName, WSName, Ext
  FROM Workers, Deans;
```


WFaculty	WFName	WSName	Ext
Me	Olaf	Ciszak	2360
Me	Olaf	Ciszak	2360
Me	Olaf	Ciszak	2360
Me	Andrzej	Milecki	2370
Me	Andrzej	Milecki	2370
Me	Andrzej	Milecki	2370
Ph	Ryszard	Czajka	3200
Ph	Ryszard	Czajka	3200
Ph	Ryszard	Czajka	3200
Ph	Ryszard	Skwarek	3210
Ph	Ryszard	Skwarek	3210
Ph	Ryszard	Skwarek	3210
Co	Ryszard	Adamiak	3410
Co	Ryszard	Adamiak	3410
Co	Ryszard	Adamiak	3410
Co	Andrzej	Jaszkiewic	3420
Co	Andrzej	Jaszkiewic	3420
Co	Andrzej	Jaszkiewic	3420
Co	Andrzej	Marciniak	3430
Co	Andrzej	Marciniak	3430
Co	Andrzej	Marciniak	3430
Co	Andrzej	Urbaniak	3440
Co	Andrzej	Urbaniak	3440
Co	Andrzej	Urbaniak	3440

Adding dean's surname

```
SELECT WFaculty, WFName, WSName, Ext, DSName
  FROM Workers, Deans;
```


WFaculty	WFName	WSName	Ext	DSName
Me	Olaf	Ciszak	2360	Jaszkiewicz
Me	Olaf	Ciszak	2360	Ciszak
Me	Olaf	Ciszak	2360	Czajka
Me	Andrzej	Milecki	2370	Jaszkiewicz
Me	Andrzej	Milecki	2370	Ciszak
Me	Andrzej	Milecki	2370	Czajka
Ph	Ryszard	Czajka	3200	Jaszkiewicz
Ph	Ryszard	Czajka	3200	Ciszak
Ph	Ryszard	Czajka	3200	Czajka
Ph	Ryszard	Skwarek	3210	Jaszkiewicz
Ph	Ryszard	Skwarek	3210	Ciszak
Ph	Ryszard	Skwarek	3210	Czajka
Co	Ryszard	Adamiak	3410	Jaszkiewicz
Co	Ryszard	Adamiak	3410	Ciszak
Co	Ryszard	Adamiak	3410	Czajka
Co	Andrzej	Jaszkiewic	3420	Jaszkiewicz
Co	Andrzej	Jaszkiewic	3420	Ciszak
Co	Andrzej	Jaszkiewic	3420	Czajka
Co	Andrzej	Marciniak	3430	Jaszkiewicz
Co	Andrzej	Marciniak	3430	Ciszak
Co	Andrzej	Marciniak	3430	Czajka
Co	Andrzej	Urbaniak	3440	Jaszkiewicz
Co	Andrzej	Urbaniak	3440	Ciszak
Co	Andrzej	Urbaniak	3440	Czajka

WFaculty	WFName	WSName	Ext	DSName
Me	Olaf	Ciszak	2360	Jaszkiewicz
Me	Olaf	Ciszak	2360	Ciszak
Me	Olaf	Ciszak	2360	Czajka
Me	Andrzej	Milecki	2370	Jaszkiewicz
Me	Andrzej	Milecki	2370	Ciszak
Me	Andrzej	Milecki	2370	Czajka
Ph	Ryszard	Czajka	3200	Jaszkiewicz
Ph	Ryszard	Czajka	3200	Ciszak
Ph	Ryszard	Czajka	3200	Czajka
Ph	Ryszard	Skwarek	3210	Jaszkiewicz
Ph	Ryszard	Skwarek	3210	Ciszak
Ph	Ryszard	Skwarek	3210	Czajka
Co	Ryszard	Adamiak	3410	Jaszkiewicz
Co	Ryszard	Adamiak	3410	Ciszak
Co	Ryszard	Adamiak	3410	Czajka
Co	Andrzej	Jaszkiewic	3420	Jaszkiewicz
Co	Andrzej	Jaszkiewic	3420	Ciszak
Co	Andrzej	Jaszkiewic	3420	Czajka
Co	Andrzej	Marciniak	3430	Jaszkiewicz
Co	Andrzej	Marciniak	3430	Ciszak
Co	Andrzej	Marciniak	3430	Czajka
Co	Andrzej	Urbaniak	3440	Jaszkiewicz

```
SELECT ... WHERE WSName = DSName;
```


Result

```
SELECT WFName, WSName, Ext
  FROM Workers, Deans
 WHERE WSName   = DSName;
```

WFName	WSName	Ext
Olaf	Ciszak	2360
Ryszard	Czajka	3200
Andrzej	Jaszkiewic	3420

Keys

Workers

WFaculty	WFName	WSName	Ext
Me	Olaf	Ciszak	2360
Me	Andrzej	Milecki	2370
Ph	Ryszard	Czajka	3200
Ph	Ryszard	Skwarek	3210
Co	Ryszard	Adamiak	3410
Co	Andrzej	Jaszkiewic	3420
Co	Andrzej	Marciniak	3430
Co	Andrzej	Urbaniak	3440

Deans

DFaculty	DFName	DSName
Co	Andrzej	Jaszkiewicz
Me	Olaf	Ciszak
Ph	Ryszard	Czajka

Problem: Repeated data

Workers

WFaculty	WFName	WSName	Ext
Me	Olaf	Ciszak	2360
Me	Andrzej	Milecki	2370
Ph	Ryszard	Czajka	3200
Ph	Ryszard	Skwarek	3210
Co	Ryszard	Adamiak	3410
Co	Andrzej	Jaszkiewic	3420
Co	Andrzej	Marciniak	3430
Co	Andrzej	Urbaniak	3440

Deans

ID
3420
2360
3200

A better version


```
SELECT WFName, WSName, Ext
  FROM Workers, Deans
 WHERE Ext = ID;
```

WFName	WSName	Ext
Olaf	Ciszak	2360
Ryszard	Czajka	3200
Andrzej	Jaszkiewic	3420

Workers

WFaculty	WFName	WSName	Ext
Me	Olaf	Ciszak	2360
Me	Andrzej	Milecki	2370
Ph	Ryszard	Czajka	3200
Ph	Ryszard	Skwarek	3210
Co	Ryszard	Adamiak	3410
Co	Andrzej	Jaszkiewic	3420
Co	Andrzej	Marciniak	3430
Co	Andrzej	Urbaniak	3440

Deans

ID
3420
2360
3200

Keys (unique values)

Riddle

```
.header on
.mode column
CREATE TABLE Workers
( WFaculty VARCHAR(10),  /* Worker's faculty       */
  /* Co = Computing, Me = Mechanics, Ph = Physics */
  WFName   VARCHAR(40), /* Worker's first Name    */
  WSName   VARCHAR(40), /* Worker's surname       */
  Ext      INT        ); /* Phone number extension */
INSERT INTO Workers VALUES ('Me', 'Olaf'    , 'Ciszak',       2360);
INSERT INTO Workers VALUES ('Me', 'Andrzej', 'Milecki',      2370);
INSERT INTO Workers VALUES ('Ph', 'Ryszard', 'Czajka',       3200);
INSERT INTO Workers VALUES ('Ph', 'Ryszard', 'Skwarek',      3210);
INSERT INTO Workers VALUES ('Co', 'Ryszard', 'Adamiak',      3410);
INSERT INTO Workers VALUES ('Co', 'Andrzej', 'Jaszkiewicz', 3420);
INSERT INTO Workers VALUES ('Co', 'Andrzej', 'Marciniak',    3430);
INSERT INTO Workers VALUES ('Co', 'Andrzej', 'Urbaniak',     3440);
CREATE TABLE Deans
( DFaculty VARCHAR(10),  /* Dean's faculty      */
  DFName   VARCHAR(40),  /* Dean's first Name */
  DSName   VARCHAR(40)); /* Dean's surname     */
INSERT INTO Deans VALUES ('Co', 'Andrzej', 'Jaszkiewicz');
INSERT INTO Deans VALUES ('Me', 'Olaf'    , 'Ciszak'     );
INSERT INTO Deans VALUES ('Ph', 'Ryszard', 'Czajka'     );
SELECT WFaculty, WFName, WSName FROM Workers, Deans
 WHERE WFName = DFName AND WFaculty = DFaculty;
```


Result

```
SELECT WFaculty, WFName, WSName
  FROM Workers, Deans
 WHERE WFName   = DFName
   AND WFaculty = DFaculty;
```

WFaculty	WFName	WSName
Me	Olaf	Ciszak
Ph	Ryszard	Czajka
Ph	Ryszard	Skwarek
Co	Andrzej	Jaszkiewic
Co	Andrzej	Marciniak
Co	Andrzej	Urbaniak

How to get all the above people but deans?

Solution

```
SELECT WFaculty, WFName, WSName
  FROM Workers, Deans
 WHERE WFName   = DFName
   AND WFaculty = DFaculty
   AND WSName  != DSName;
```

WFaculty	WFName	WSName
Ph	Ryszard	Skwarek
Co	Andrzej	Marciniak
Co	Andrzej	Urbaniak

Aggregation functions

- COUNT(*col*)
- MIN(*col*)
- MAX(*col*)
- SUM(*col*)
- AVG(*col*)

Aggregation functions

Workers

Key	Hours	SName
20	170	Ciszak
21	90	Miklosik
22	140	Nawrocki
23	160	Pacholski

```
SELECT COUNT(SName)
  FROM Workers;
```

```
count
-------
      4
```


Aggregation functions

Workers

Key	Hours	SName
20	170	Ciszak
21	90	Miklosik
22	140	Nawrocki
23	160	Pacholski

```
SELECT COUNT(*)
  FROM Workers;
```

```
count
-------
      4
```


Aggregation functions

- COUNT(*col*)
- MIN(*col*)
- MAX(*col*)
- SUM(*col*)
- AVG(*col*)

SQL – Basic operations

CREATE an empty table with a given set of columns
INSERT a row into a given table
SELECT (read) data from table(s)
UPDATE some rows of a given table
DELETE some rows of a given table

Updating data

Workers

Key	FName	SName
20	Konrad	Ciszak
21	Piotr	Miklosik
22	Konrad	Nawrocki
23	Leszek	Pacholski

Jerzy, not Konrad

```
UPDATE Workers
  SET FName= 'Jerzy'
  WHERE SName= 'Nawrocki';
```

Updating data

Workers

Key	FName	SName
20	Konrad	Ciszak
21	Piotr	Miklosik
22	Jerzy	Nawrocki
23	Leszek	Pacholski

Jerzy, not Konrad

```
UPDATE Workers
   SET FName= 'Jerzy'
   WHERE SName= 'Nawrocki';
```


SQL – Basic operations

CREATE an empty table with a given set of columns
INSERT a row into a given table
SELECT (read) data from table(s)
UPDATE some rows of a given table
DELETE some rows of a given table

Deleting data

Workers

Key	FName	SName
20	Konrad	Ciszak
21	Piotr	Miklosik
22	Konrad	Nawrocki
23	Leszek	Pacholski

```
DELETE FROM Workers
   WHERE FName= 'Konrad';
```

Key	FName	SName
21	Piotr	Miklosik
23	Leszek	Pacholski

Agenda

- File processing
- Relational databases
- Basics of SQL
- **Advanced features of SQL**
- Python interface to SQLite3
- Machine learning & ID3


```
3  CREATE TABLE Workers
4  ( WFaculty VARCHAR(10),  /* Worker's faculty       */
5    /* Co = Computing, Me = Mechanics, Ph = Physics */
6    WFName   VARCHAR(40), /* Worker's first Name     */
7    WSName   VARCHAR(40), /* Worker's surname        */
8    Ext      INT        ); /* Phone number extension */
9  INSERT INTO Workers VALUES ('Me', 'Olaf'    , 'Ciszak',     2360);
10 INSERT INTO Workers VALUES ('Me', 'Andrzej', 'Milecki',    2370);
11 INSERT INTO Workers VALUES ('Ph', 'Ryszard', 'Czajka',     3200);
12 INSERT INTO Workers VALUES ('Ph', 'Ryszard', 'Skwarek',    3210);
13 INSERT INTO Workers VALUES ('Co', 'Ryszard', 'Adamiak',    3410);
14 INSERT INTO Workers VALUES ('Co', 'Andrzej', 'Jaszkiewicz',3420);
15 INSERT INTO Workers VALUES ('Co', 'Andrzej', 'Marciniak',  3430);
16 INSERT INTO Workers VALUES ('Co', 'Andrzej', 'Urbaniak',   3440);
17
18 SELECT WFaculty, COUNT(*) FROM Workers GROUP BY WFaculty;
```

WFaculty	COUNT(*)
Co	4
Me	2
Ph	2

Group of rows: the same value of a given attrib.

Distinct items

Workers

Key	FName	SName
20	Konrad	Ciszak
21	Piotr	Miklosik
22	Konrad	Nawrocki
23	Leszek	Pacholski

First names of the workers?

```
SELECT FName
   FROM Workers
   ORDER BY FName ASC;
```

FName
Konrad
Konrad
Leszek
Piotr

Distinct items

Workers

Key	FName	SName
20	Konrad	Ciszak
21	Piotr	Miklosik
22	Konrad	Nawrocki
23	Leszek	Pacholski

First names of the workers?

```
SELECT DISTINCT FName
    FROM Workers
    ORDER BY FName ASC;
```

FName
Konrad
Leszek
Piotr

Patterns

Workers

Key	FName	SName
20	Konrad	Ciszak
21	Piotr	Miklosik
22	Konrad	Nawrocki
23	Leszek	Pacholski

```
SELECT FName, SName
    FROM Workers
    WHERE SName LIKE '%ki';
```

FName	SName
Konrad	Nawrocki
Leszek	Pacholski

Patterns

Workers

Key	FName	SName
20	Konrad	Ciszak
21	Piotr	Miklosik
22	Konrad	Nawrocki
23	Leszek	Pacholski

```
SELECT FName, SName
    FROM Workers
    WHERE SName LIKE '_a%';
```

FName	SName
Konrad	Nawrocki
Leszek	Pacholski

Storing results in a table

Workers

Key	FName	SName
20	Konrad	Ciszak
21	Piotr	Miklosik
22	Konrad	Nawrocki
23	Leszek	Pacholski

First names of the workers

```
CREATE TABLE WorkNams
(
  Name VARCHAR(40),
);

INSERT INTO WorkNams
SELECT DISTINCT FName
    FROM Workers
    ORDER BY FName ASC;
```

WorkNams

Name
Konrad
Leszek
Piotr

Nested queries

Workers

Key	Hours	SName
20	170	Ciszak
21	90	Miklosik
22	140	Nawrocki
23	160	Pacholski

```
SELECT * FROM Workers
WHERE Hours > (SELECT AVG(Hours) FROM Workers);
```

Key	Hours	SName
20	170	Ciszak
23	160	Pacholski

Agenda

- File processing
- Relational databases
- Basics of SQL
- Advanced features of SQL
- **Python interface to SQLite3**
- Machine learning & ID3

Mute vs. talkative statements

CREATE an empty table with a given set of columns
INSERT a row into a given table
UPDATE some rows of a given table
DELETE some rows of a given table

SELECT (read) data from table(s)

Python interface to SQLite

```
import sqlite3
db = sqlite3.connect('fileName')
c = db.cursor()
c.execute("SQLstatement")
db.commit()
db.close()
```


Example 1 (mute statements only)

```
1  CREATE TABLE Workers
2  (FName VARCHAR(40), /* First Name */
3   SName VARCHAR(40)  /* Surname    */
4  );
5  INSERT INTO Workers VALUES ('Olaf' , 'Ciszak');
6  INSERT INTO Workers VALUES ('Piotr', 'Miklosik');
```

Workers

FName	SName
Olaf	Ciszak
Piotr	Miklosik

Using 'mute' statements

```
import sqlite3
db = sqlite3.connect('my.db')
c = db.cursor()
c.execute('''create table Workers
             (FName VARCHAR(40),
              SName VARCHAR(40)
             );''')
c.execute("insert into Workers values('Olaf', 'Ciszak'  );")
c.execute("insert into Workers values('Piotr','Miklosik');")
db.commit()
db.close()
```


Talkative statements

CREATE an empty table with a given set of columns
INSERT a row into a given table
UPDATE some rows of a given table
DELETE some rows of a given table

SELECT (read) data from table(s)

Using 'talkative' statements

```
import sqlite3
db = sqlite3.connect('my.db')
c = db.cursor()
c.execute('''create table Workers
             (FName VARCHAR(40),
              SName VARCHAR(40)
             );''')
c.execute("insert into Workers values('Olaf', 'Ciszak'  );")
c.execute("insert into Workers values('Piotr','Miklosik');")
db.commit()
for row in c.execute("select SName from Workers;"):
    print(row)
db.close()
```

```
('Ciszak',)
('Miklosik',)
```

Using 'talkative' statements

```
import sqlite3
db = sqlite3.connect('my.db')
c = db.cursor()
c.execute('''create table Workers
             (FName VARCHAR(40),
              SName VARCHAR(40)
             );''')
c.execute("insert into Workers values('Olaf', 'Ciszak'  );")
c.execute("insert into Workers values('Piotr','Miklosik');")
db.commit()
for row in c.execute("select SName from Workers;"):
    print(row[0])
db.close()
```

```
Ciszak
Miklosik
```


The three basic functions

C

```
    sqlite3_open(                 );
...
    sqlite3_exec(                                 );
...
    sqlite3_close(   );
```


Database pointer

C

```
#include <sqlite3.h>
...
sqlite3 *db;

    sqlite3_open(fileName, &db);
...
    sqlite3_exec(db,                          );
...
    sqlite3_close(db);
```


Executing mute statements

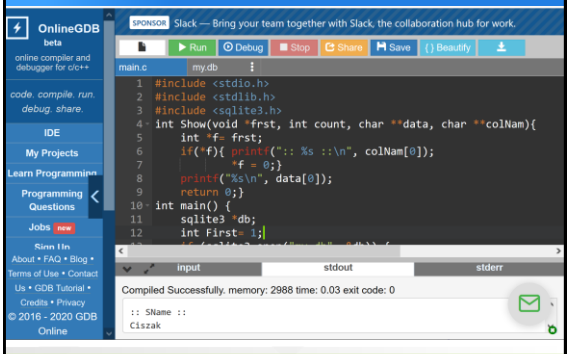
C

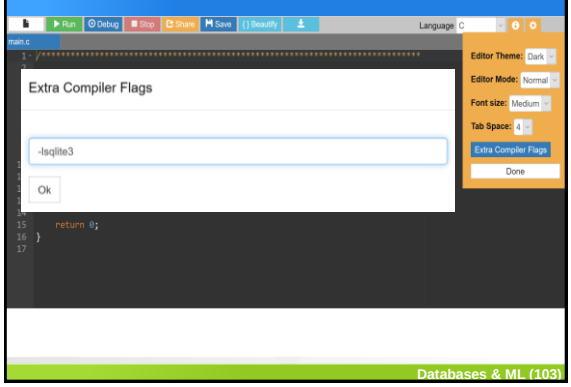
```
#include <sqlite3.h>
...
sqlite3 *db;
...
Err= sqlite3_open(fileName, &db);
...
Err= sqlite3_exec(db, query, NULL, NULL, NULL);
...
Err= sqlite3_close(db);
```


C

```
#include <stdio.h>
#include <stdlib.h>
#include <sqlite3.h>
int main() {
    sqlite3 *db;
    if (sqlite3_open("my.db", &db)) {
       fprintf(stderr, "Could not open my.db\n");
       exit(-1); }
    if (sqlite3_exec(db,
        "create table Workers("
            "FName VARCHAR(40),"
            "SName VARCHAR(40));",
        NULL, NULL, NULL))
        fprintf(stderr,"Error executing CREATE\n");
    if (sqlite3_exec(db,
        "insert into Workers values('Olaf', 'Ciszak');"
        "insert into Workers values('Piotr','Miklosik');",
        NULL, NULL, NULL))
        fprintf(stderr, "Error executing INSERT\n");
    sqlite3_close(db);
    return 0;  }
```


https://www.onlinegdb.com/online_c_compiler





Agenda

- File processing
- Relational databases
- Basics of SQL
- Advanced features of SQL
- Python interface to SQLite3
- **Machine learning & ID3**

Machine learning (ML)

ML = the study of computer algorithms that can improve automatically through experience and by the use of data

(-) Tom M. Mitchell

https://www.cmu.edu/bme/People/Faculty/profile/tmitchell.html

Main categories of ML

Learning decision making from examples

Identification of data clusters

Playing a game and learning from feedback

- **Supervised learning**
- Unsupervised learning
- Reinforcement learning

Supervised learning ≈ Learning from examples

Labels

Training data

Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Model of decision making

- **Decision tree**
- **Decision table**
- **Neural network**
- . . .

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

Supervised learning – ID3

- **Iterative Dichotomizer 3 (ID3)**
- **C 4.5**

Ross Quinlan

Dichotomy = a division or contrast between two opposed things

Supervised learning ≈ Learning from examples

Training data

Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

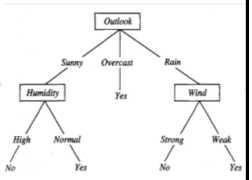
→ **Model of decision making**

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

ID3 – General idea

Training data

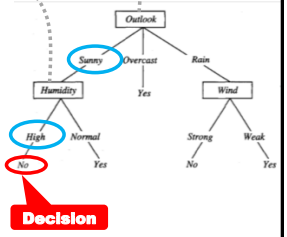
Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

→ 

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

Decision trees

Outlook	Temp	Humidity	Wind
Sunny	Hot	High	Weak



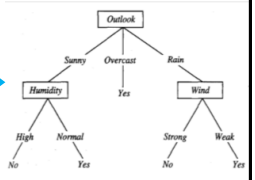
Decision

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

ID3 – General idea

Training data

Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

→ ? → 

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

ID3 – General idea

Training data

Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

1. **If the data are unanimous then STOP.**
2. **If not: Which column to choose?**
3. **What are the values in the column?**
4. **Divide the table into subtables by those values.**
5. **Repeat the procedure for each subtable.**

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

ID3 – General idea

Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

1. **If the data are unanimous then STOP.**
2. **If not: Which column to choose?**
3. **What are the values in the column?**
4. **Divide the table into subtables by those values.**
5. **Repeat the procedure for each subtable.**

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

ID3 – General idea

Overcast
Rain
Sunny

Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

1. If the data are unanimous then STOP.
2. If not: Which column to choose?
3. What are the values in the column?
4. Divide the table into subtables by those values.
5. Repeat the procedure for each subtable.

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

ID3 – General idea

Overcast
Rain
Sunny

Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

1. If the data are unanimous then STOP.
2. If not: Which column to choose?
3. What are the values in the column?
4. **Divide the table into subtables by those values.**
5. Repeat the procedure for each subtable.

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

Examplary decision tree (T.M. Mitchell 1997)

Outlook = Overcast

Temp	Humidity	Wind	Opinion
Hot	High	Weak	Yes

Day*	Outlook	Temp	Humidity	Wind	Opinion
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Examplary decision tree (T.M. Mitchell 1997)

Outlook = Overcast

Temp	Humidity	Wind	Opinion
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

Day*	Outlook	Temp	Humidity	Wind	Opinion
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Examplary decision tree (T.M. Mitchell 1997)

Outlook = Overcast

Temp	Humidity	Wind	Opinion
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

Outlook = Rain

Temp	Humidity	Wind	Opinion
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No

Outlook = Sunny

Temp	Humidity	Wind	Opinion
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

Examplary decision tree (T.M. Mitchell 1997)

Outlook = Overcast

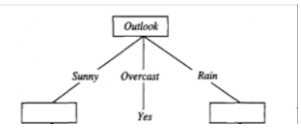
Temp	Humidity	Wind	Opinion
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

Outlook = Rain

Temp	Humidity	Wind	Opinion
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No

Outlook = Sunny

Temp	Humidity	Wind	Opinion
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes



Examplary decision tree (T.M. Mitchell 1997)

Outlook = Overcast

Temp	Humidity	Wind	Opinion
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

Outlook = Rain

Temp	Humidity	Wind	Opinion
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No

Outlook = Sunny

Temp	Humidity	Wind	Opinion
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

1. If the data are unanimous then STOP.
2. If not: Which column to choose?
3. What are the values in the column?
4. Divide the table into subtables by those values.
5. **Repeat the procedure for each subtable.**

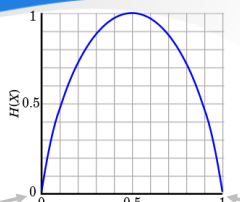
ID3 – General idea

Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

1. If the data are unanimous then STOP.
2. If not: **Which column to choose?**
3. What are the values in the column?
4. Divide the table into subtables by those values.
5. Repeat the procedure for each subtable.

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

Entropy (Claude Shannon)



$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$

https://en.wikipedia.org/wiki/Entropy_(information_theory)

Examplary decision tree (T.M. Mitchell 1997)

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$

Day*	Outlook	Temp	Humidity	Wind	Opinion
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

	Frq	P	-P log2 P
Yes	9	0.64	0.41
No	5	0.36	0.53
	14		0.94

Entropy

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

Examplary decision tree (T.M. Mitchell 1997)

Day*	Outlook	Temp	Humidity	Wind	Opinion
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

Examplary decision tree (T.M. Mitchell 1997)

Day*	Outlook	Temp	Humidity	Wind	Opinion
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

Examplary decision tree (T.M. Mitchell 1997)

Day*	Outlook	Temp	Humidity	Wind	Opinion
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

Examplary decision tree (T.M. Mitchell 1997)

Day*	Outlook	Temp	Humidity	Wind	Opinion
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Examplary decision tree (T.M. Mitchell 1997)

Day*	Outlook	Temp	Humidity	Wind	Opinion
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Overcast
Rain
Sunny

Examplary decision tree (T.M. Mitchell 1997)

Outlook = Overcast

Temp	Humidity	Wind	Opinion
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

Outlook = Rain

Temp	Humidity	Wind	Opinion
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No

Outlook = Sunny

Temp	Humidity	Wind	Opinion
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

Examplary decision tree (T.M. Mitchell 1997)

Outlook = Overcast

Temp	Humidity	Wind	Opinion
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

H = Entropy

H(Overcast) = 0 (all decisions the same)

Outlook = Rain

Temp	Humidity	Wind	Opinion
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No

	Frq	P	-P log2 P
Yes	3	0.60	0.44
No	2	0.40	0.53
	5		0.97

H(Rain) = 0.97

Outlook = Sunny

Temp	Humidity	Wind	Opinion
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

	Frq	P	-P log2 P
Yes	2	0.40	0.53
No	3	0.60	0.44
	5		0.97

H(Sunny) = 0.97

Examplary decision tree (T.M. Mitchell 1997)

Outlook = Overcast

Temp	Humidity	Wind	Opinion
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

H(Overcast) = 0
P(Overcast) = 4 / 14 ≈ 0.29

Outlook = Rain

Temp	Humidity	Wind	Opinion
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No

H(Rain) = 0.97
P(Rain) = 5 / 14 ≈ 0.36

Outlook = Sunny

Temp	Humidity	Wind	Opinion
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

H(Sunny) = 0.97
P(Sunny) = 5 / 14 ≈ 0.36

Examplary decision tree (T.M. Mitchell 1997)

Outlook = Overcast

Temp	Humidity	Wind	Opinion
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

H(Overcast) = 0
P(Overcast) ≈ 0.29

Outlook = Rain

Temp	Humidity	Wind	Opinion
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No

H(Rain) = 0.97
P(Rain) ≈ 0.36

Outlook = Sunny

Temp	Humidity	Wind	Opinion
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

H(Sunny) = 0.97
P(Sunny) ≈ 0.36

H(Outlook) =
P(Overcast) * H(Overcast)
+ P(Rain) * H(Rain)
+ P(Sunny) * H(Sunny) ≈
0.29*0 + 0.36*0.97 + 0.36*0.97
≈ 0.69

ID3 (R. Quinlan, 1986)

Which column to choose?

0.69	0.91	0.79	0.89	
Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

c = selected column
G(c) = information gain for **c**

G(c) = H(table) – H(c)

Column selection rule:
maximize G(c)

In other words:
minimize H(c)

ID3 – General idea

Outlook	Temp	Humidity	Wind	Opinion
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

1. If the data are unanimous then STOP.
2. If not: **Which column to choose?**
3. What are the values in the column?
4. Divide the table into subtables by those values.
5. Repeat the procedure for each subtable.

https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/

Agenda

Summary

Your feedback is important!

9. Databases & Machine Learning

Your opinion on Lecture 9 (Databases & ML)

3 simple questions + your remarks

Individual test

December 21 2023 (Thursday), 16:50 – 18:20
Lecture rooms: CW-4, CW-8, CW-11

- **Open-book test**
- **Taking the test is optional (but highly recommended):**
  Guaranteed grade if score > 50% else No Guarantee
- **Participation in the contest without taking the test:**
  Grade from the contest depends on February exam