Bazy danych 2024

20 lutego 2024

Plan

Po co nam bazy danych? Do czego?

- Pozwalają na abstrakcję (nieważne co w środku!) - logiczną/fizyczną,

Po co nam bazy danych? Do czego?

- Pozwalają na abstrakcję (nieważne co w środku!) - logiczną/fizyczną,
- Uwalniają twórcę systemu od implementowania warstwy danych w każdym projekcie,

Po co nam bazy danych? Do czego?

- Pozwalają na abstrakcję (nieważne co w środku!) - logiczną/fizyczną,
- Uwalniają twórcę systemu od implementowania warstwy danych w każdym projekcie,
- Zapewniają integralność danych (ACID)

Po co nam bazy danych? Do czego?

- Pozwalają na abstrakcję (nieważne co w środku!) - logiczną/fizyczną,
- Uwalniają twórcę systemu od implementowania warstwy danych w każdym projekcie,
- Zapewniają integralność danych (ACID)

  atomowość (**Atomicity**), poprawność (**Consistency**), niezależność (**Isolation**), trwałość (**Durability**).

Po co nam bazy danych? Do czego?

- Pozwalają na abstrakcję (nieważne co w środku!) - logiczną/fizyczną,
- Uwalniają twórcę systemu od implementowania warstwy danych w każdym projekcie,
- Zapewniają integralność danych (ACID)

  atomowość (**Atomicity**), poprawność (**Consistency**), niezależność (**Isolation**), trwałość (**Durability**).
- Zapewniają security (kontrolę dostępu),

Po co nam bazy danych? Do czego?

- Pozwalają na abstrakcję (nieważne co w środku!) - logiczną/fizyczną,
- Uwalniają twórcę systemu od implementowania warstwy danych w każdym projekcie,
- Zapewniają integralność danych (ACID)

  atomowość (**Atomicity**), poprawność (**Consistency**), niezależność (**Isolation**), trwałość (**Durability**).
- Zapewniają security (kontrolę dostępu),
- Pozwalają na optymalizacje.

Po co nam bazy danych? Do czego?

- Pozwalają na abstrakcję (nieważne co w środku!) - logiczną/fizyczną,
- Uwalniają twórcę systemu od implementowania warstwy danych w każdym projekcie,
- Zapewniają integralność danych (ACID)

  atomowość (**Atomicity**), poprawność (**Consistency**), niezależność (**Isolation**), trwałość (**Durability**).
- Zapewniają security (kontrolę dostępu),
- Pozwalają na optymalizacje.

Po co nam bazy danych? Do czego?

- Pozwalają na abstrakcję (nieważne co w środku!) - logiczną/fizyczną,
- Uwalniają twórcę systemu od implementowania warstwy danych w każdym projekcie,
- Zapewniają integralność danych (ACID)

  atomowość (**Atomicity**), poprawność (**Consistency**), niezależność (**Isolation**), trwałość (**Durability**).
- Zapewniają security (kontrolę dostępu),
- Pozwalają na optymalizacje.

- OLTP (duża liczba małych **transakcji**, scale-independence),
- OLAP (analiza danych, zapytania przetwarzają duże dane, ML-in-database).

Object–relational mapping (ORM)

- pozwala na serializację i deserializację obiektów do bazy danych z poziomu języka programowania,

Object–relational mapping (ORM)

- pozwala na serializację i deserializację obiektów do bazy danych z poziomu języka programowania, magia!

Object–relational mapping (ORM)

- pozwala na serializację i deserializację obiektów do bazy danych z poziomu języka programowania, magia!
- wprowadza warstwę abstrakcji - nie musisz martwić się pisaniem SQL - tworzysz obiekty i już,

Object–relational mapping (ORM)

- pozwala na serializację i deserializację obiektów do bazy danych z poziomu języka programowania, magia!
- wprowadza warstwę abstrakcji - nie musisz martwić się pisaniem SQL - tworzysz obiekty i już,
- konwertuje niezgodne typy,

Object–relational mapping (ORM)

- pozwala na serializację i deserializację obiektów do bazy danych z poziomu języka programowania, magia!
- wprowadza warstwę abstrakcji - nie musisz martwić się pisaniem SQL - tworzysz obiekty i już,
- konwertuje niezgodne typy,
- zmniejsza ilość kodu, łatwiej go utrzymywać (*separation of concerns*)

Object–relational mapping (ORM)

- pozwala na serializację i deserializację obiektów do bazy danych z poziomu języka programowania, magia!
- wprowadza warstwę abstrakcji - nie musisz martwić się pisaniem SQL - tworzysz obiekty i już,
- konwertuje niezgodne typy,
- zmniejsza ilość kodu, łatwiej go utrzymywać (*separation of concerns*)
- database-agnostic - możesz podmienić bazę na inną (a sam ORM możesz???),

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler
- **Problematyczny** gdy aplikacja jest wymaga tuningu wydajnościowego albo zapytania są skomplikowane.

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler
- **Problematyczny** gdy aplikacja jest wymaga tuningu wydajnościowego albo zapytania są skomplikowane.
  - może być wolniejsze,

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler
- **Problematyczny** gdy aplikacja jest wymaga tuningu wydajnościowego albo zapytania są skomplikowane.
  - może być wolniejsze,
  - każdy ORM jest inny,

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler
- **Problematyczny** gdy aplikacja jest wymaga tuningu wydajnościowego albo zapytania są skomplikowane.
  - może być wolniejsze,
  - każdy ORM jest inny,
  - może nie obsługiwać tego co potrzebujesz (np. dane przestrzenne),

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler
- **Problematyczny** gdy aplikacja jest wymaga tuningu wydajnościowego albo zapytania są skomplikowane.
  - może być wolniejsze,
  - każdy ORM jest inny,
  - może nie obsługiwać tego co potrzebujesz (np. dane przestrzenne),
  - trudniej debugować,

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler
- **Problematyczny** gdy aplikacja jest wymaga tuningu wydajnościowego albo zapytania są skomplikowane.
  - może być wolniejsze,
  - każdy ORM jest inny,
  - może nie obsługiwać tego co potrzebujesz (np. dane przestrzenne),
  - trudniej debugować,
  - złe pokusy (historyjka o współbieżności).

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler
- **Problematyczny** gdy aplikacja jest wymaga tuningu wydajnościowego albo zapytania są skomplikowane.
  - może być wolniejsze,
  - każdy ORM jest inny,
  - może nie obsługiwać tego co potrzebujesz (np. dane przestrzenne),
  - trudniej debugować,
  - złe pokusy (historyjka o współbieżności).

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler
- **Problematyczny** gdy aplikacja jest wymaga tuningu wydajnościowego albo zapytania są skomplikowane.
  - może być wolniejsze,
  - każdy ORM jest inny,
  - może nie obsługiwać tego co potrzebujesz (np. dane przestrzenne),
  - trudniej debugować,
  - złe pokusy (historyjka o współbieżności).

```
employee = query_all();
for (emp in employee) {
    remuneration = query_rem(emp);
   // ...
}
```

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler
- **Problematyczny** gdy aplikacja jest wymaga tuningu wydajnościowego albo zapytania są skomplikowane.
  - może być wolniejsze,
  - każdy ORM jest inny,
  - może nie obsługiwać tego co potrzebujesz (np. dane przestrzenne),
  - trudniej debugować,
  - złe pokusy (historyjka o współbieżności).

```
employee = query_all();
for (emp in employee) {
    remuneration = query_rem(emp);
   // ...
}
```

```
SELECT * FROM employee WHERE ...
SELECT * FROM rem WHERE empID = 1
SELECT * FROM rem WHERE empID = 2
SELECT * FROM rem WHERE empID = 3
SELECT * FROM rem WHERE empID = 4
...
```

Object–relational mapping (ORM)

- **Rewelacyjny** w typowych przypadkach - *Essentially the ORM can handle about 80-90% of the mapping problems, but that last chunk always needs careful work by somebody who really understands how a relational database works.* M.Fowler
- **Problematyczny** gdy aplikacja jest wymaga tuningu wydajnościowego albo zapytania są skomplikowane.
  - może być wolniejsze,
  - każdy ORM jest inny,
  - może nie obsługiwać tego co potrzebujesz (np. dane przestrzenne),
  - trudniej debugować,
  - złe pokusy (historyjka o współbieżności).

```
employee = query_all();
for (emp in employee) {
    remuneration = query_rem(emp);
   // ...
}
```

```
SELECT * FROM employee WHERE ...
SELECT * FROM rem WHERE empID = 1
SELECT * FROM rem WHERE empID = 2
SELECT * FROM rem WHERE empID = 3
SELECT * FROM rem WHERE empID = 4
...
```

Linki dla zainteresowanych:
n+1 select
ORM Hate

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite),

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model
- klucz-wartość (np. Redis),

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model
- klucz-wartość (np. Redis),
- dokumentowe (np. MongoDB),

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model
- klucz-wartość (np. Redis),
- dokumentowe (np. MongoDB),
- grafowe (np. Neo4j),

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model
- klucz-wartość (np. Redis),
- dokumentowe (np. MongoDB),
- grafowe (np. Neo4j),
- wyszukiwarki (full-text search, np. Elasticsearch, Splunk, SOLR),

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model
- klucz-wartość (np. Redis),
- dokumentowe (np. MongoDB),
- grafowe (np. Neo4j),
- wyszukiwarki (full-text search, np. Elasticsearch, Splunk, SOLR),
- szerokokolumnowe? (wide-column stores, np. Cassandra),

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model
- klucz-wartość (np. Redis),
- dokumentowe (np. MongoDB),
- grafowe (np. Neo4j),
- wyszukiwarki (full-text search, np. Elasticsearch, Splunk, SOLR),
- szerokokolumnowe? (wide-column stores, np. Cassandra),
- przestrzenne (spatial, np. PostGIS),

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model
- klucz-wartość (np. Redis),
- dokumentowe (np. MongoDB),
- grafowe (np. Neo4j),
- wyszukiwarki (full-text search, np. Elasticsearch, Splunk, SOLR),
- szerokokolumnowe? (wide-column stores, np. Cassandra),
- przestrzenne (spatial, np. PostGIS),
- wektorowe (np. Kdb, Pinecone)

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model
- klucz-wartość (np. Redis),
- dokumentowe (np. MongoDB),
- grafowe (np. Neo4j),
- wyszukiwarki (full-text search, np. Elasticsearch, Splunk, SOLR),
- szerokokolumnowe? (wide-column stores, np. Cassandra),
- przestrzenne (spatial, np. PostGIS),
- wektorowe (np. Kdb, Pinecone)
- szeregi czasowe (np. Influx)

Typy baz danych: SQL/NoSQL

- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model
- klucz-wartość (np. Redis),
- dokumentowe (np. MongoDB),
- grafowe (np. Neo4j),
- wyszukiwarki (full-text search, np. Elasticsearch, Splunk, SOLR),
- szerokokolumnowe? (wide-column stores, np. Cassandra),
- przestrzenne (spatial, np. PostGIS),
- wektorowe (np. Kdb, Pinecone)
- szeregi czasowe (np. Influx)

Typy baz danych: SQL/NoSQL

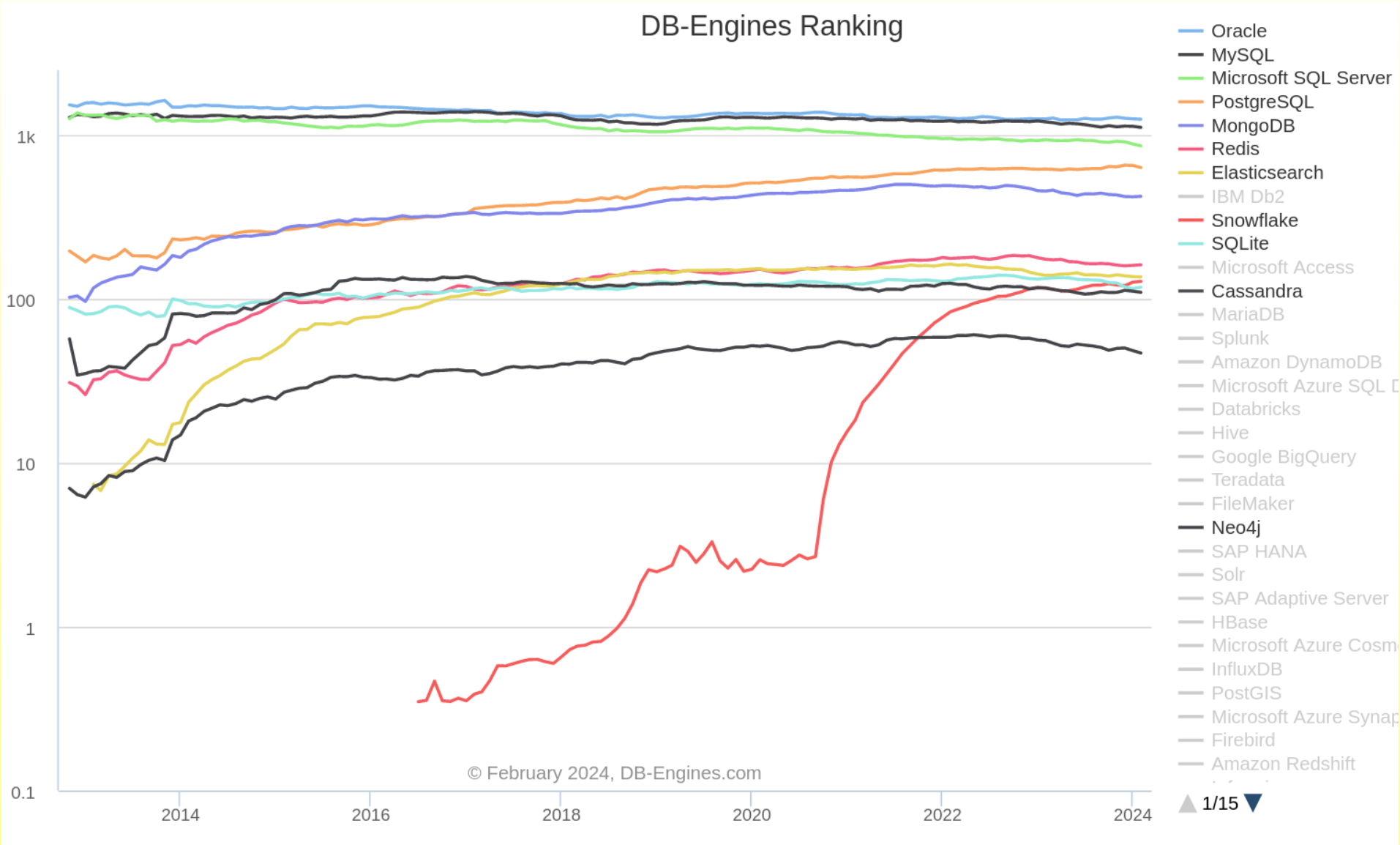
- relacyjne (np. Oracle, MySQL, SQL Server, PostgreSQL, SQLite), multi-model
- klucz-wartość (np. Redis),
- dokumentowe (np. MongoDB),
- grafowe (np. Neo4j),
- wyszukiwarki (full-text search, np. Elasticsearch, Splunk, SOLR),
- szerokokolumnowe? (wide-column stores, np. Cassandra),
- przestrzenne (spatial, np. PostGIS),
- wektorowe (np. Kdb, Pinecone)
- szeregi czasowe (np. Influx)

(. . .) we are gearing up for a shift to ***polyglot persistence*** *where any decent sized enterprise will have a variety of different data storage technologies for different kinds of data. There will still be large amounts of it managed in relational stores, but increasingly we'll be first asking how we want to manipulate the data and only then figuring out what technology is the best bet for it.* Martin Fowler

Popularność: https://db-engines.com/en/ranking

Feb 2024	Jan 2024	Feb 2023	DBMS	Database Model	Feb 2024	Jan 2024	Feb 2023
1.	1.	1.	Oracle	Relational, Multi-model	1241.45	-6.05	-6.08
2.	2.	2.	MySQL	Relational, Multi-model	1106.67	-16.79	-88.78
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	853.57	-23.03	-75.52
4.	4.	4.	PostgreSQL	Relational, Multi-model	629.41	-19.55	+12.90
5.	5.	5.	MongoDB	Document, Multi-model	420.36	+2.88	-32.41
6.	6.	6.	Redis	Key-value, Multi-model	160.71	+1.33	-13.12
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	135.74	-0.33	-2.86
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model	132.23	-0.18	-10.74
9.	9.	↑ 12.	Snowflake	Relational	127.45	+1.53	+11.80
10.	↑ 11.	↓ 9.	SQLite	Relational	117.28	+2.08	-15.38
11.	↓ 10.	↓ 10.	Microsoft Access	Relational	113.17	-4.50	-17.86
12.	12.	↓ 11.	Cassandra	Wide column, Multi-model	109.27	-1.77	-6.95
13.	13.	13.	MariaDB	Relational, Multi-model	97.23	-2.00	+0.42
14.	14.	14.	Splunk	Search engine	91.65	-1.07	+4.57
15.	↑ 16.	15.	Amazon DynamoDB	Multi-model	82.90	+1.96	+3.21
16.	↓ 15.	16.	Microsoft Azure SQL Database	Relational, Multi-model	79.56	-1.51	+0.81
17.	17.	↑ 19.	Databricks	Multi-model	76.91	-3.62	+16.58
18.	18.	↓ 17.	Hive	Relational	65.81	-1.15	-6.31
19.	19.	↑ 22.	Google BigQuery	Relational	63.63	+0.15	+11.17
20.	20.	↓ 18.	Teradata	Relational, Multi-model	51.24	-1.94	-11.79
21.	21.	21.	FileMaker	Relational	50.48	-1.56	-2.32
22.	22.	↓ 20.	Neo4j	Graph	46.61	-1.57	-8.82
23.	23.	23.	SAP HANA	Relational, Multi-model	45.22	-1.22	-4.45

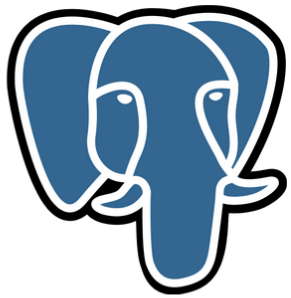
Trendy: https://db-engines.com/en/ranking_trend



Plan

Relacyjne bazy danych

Relacyjne bazy danych



Relacyjne bazy danych

"n.id"	"n.screen_name"	"n.followers_count"	"n.friends_count"	"n.location"
2531159968	"traceyhappymom"	3696	3353	"Washington, DC"
100345056	"SCOTTGOHARD"	1053	1055	"still ↑Block↴Corner↑street"
247165706	"Beckster319"	650	896	"Chicago, IL"
249538861	"skatewake1994"	44	154	""
449689677	"KadirovRussia"	94773	7	""
471868887	"MargoSavazh"	23305	8021	"Санкт-Петербург, Россия"
1039581360	"darknally"	22	40	"Amerika"
1510488662	"lagonehoe"	3080	2369	"USA"
1513801268	"YouJustCtrlC"	2760	2700	"USA"
1517678892	"MrMoraan"	879	758	"Philadelphia, PA"
1518857420	"NoJonathonNo"	789	440	"USA"

Relacyjne bazy danych

"n.id"	"n.screen_name"	"n.followers_count"	"n.friends_count"	"n.location"
2531159968	"traceyhappymom"	3696	3353	"Washington, DC"
100345056	"SCOTTGOHARD"	1053	1055	"still ıBlockıCornerıstreet"
247165706	"Beckster319"	650	896	"Chicago, IL"
249538861	"skatewak			
449689677	"KadirovF			
471868887	"MargoSav			
1039581360	"darknall			
1510488662	"lagonehc			
1513801268	"YouJustC			
1517678892	"MrMoraar			
1518857420	"NoJonath			

"n.tweet_id"	"n.user_key"	"n.created_str"	"text"
836227891897651201	"kathiemrr"	"2017-02-27 14:54:00"	"ThingsDoneByMist
765198948239810561	"traceyhappymom"	"2016-08-15 14:50:20"	"T @mc_derpin: #T
616002306572746752	"evewebster373"	"2015-06-30 21:56:09"	"T @dmataconis: R
776693302926147584	"blacktolive"	"2016-09-16 08:04:48"	"men! #blacklives
777594647875059712	"jacquelinisbest"	"2016-09-18 19:46:25"	"T @NahBabyNah: T
718040061649031168	"judelambertusa"	"2016-04-07 11:37:45"	"T @mcicero10: #B
785586729579196416	"carriethornthon"	"2016-10-10 21:04:06"	"T @ItsJustJaynie
664592113775022084	"johnbranchh"	"2015-11-11 23:54:42"	"TodayCleveland '
782408661389840384	"march_for_trump"	"2016-10-02 02:35:35"	"NickTomaWBRE Hi,
811580868573691904	"puredavie"	"2016-12-21 14:35:32"	"hat. Is. A. Reso
800655998335774720	"daileyjadon"	"2016-11-21 11:04:00"	"ifetime movie
822266545728585728	"evagreen69"	"2017-01-20 02:16:36"	"T @Conservatexia
795661161282891776	"cassieweltch"	"2016-11-07 16:16:18"	"T @HillaryClinto
777859822679158784	"_nickluna_"	"2016-09-19 13:20:08"	"T @leonpui_: Hil

Relacyjne bazy danych

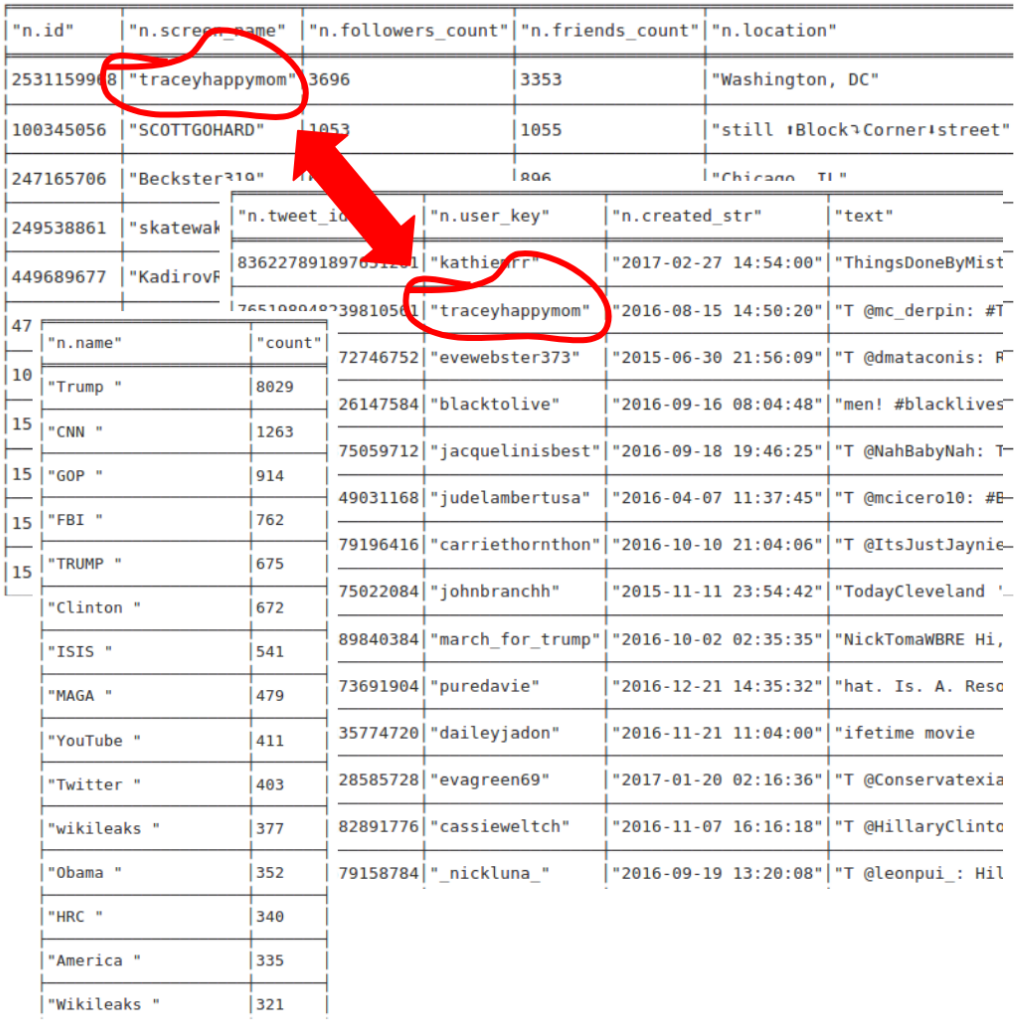
"n.id"	"n.screen_name"	"n.followers_count"	"n.friends_count"	"n.location"
2531159968	"traceyhappymom"	3696	3353	"Washington, DC"
100345056	"SCOTTGOHARD"	1053	1055	"still ıBlockıCornerıstreet"
247165706	"Beckster319"	650	896	"Chicago, IL"
249538861	"skatewak			
449689677	"KadirovF			

"n.tweet_id"	"n.user_key"	"n.created_str"	"text"
836227891897651201	"kathiemrr"	"2017-02-27 14:54:00"	"ThingsDoneByMist
765198948239810561	"traceyhappymom"	"2016-08-15 14:50:20"	"T @mc_derpin: #T
72746752	"evewebster373"	"2015-06-30 21:56:09"	"T @dmataconis: R
26147584	"blacktolive"	"2016-09-16 08:04:48"	"men! #blacklives
75059712	"jacquelinisbest"	"2016-09-18 19:46:25"	"T @NahBabyNah: T
49031168	"judelambertusa"	"2016-04-07 11:37:45"	"T @mcicero10: #B
79196416	"carriethornthon"	"2016-10-10 21:04:06"	"T @ItsJustJaynie
75022084	"johnbranchh"	"2015-11-11 23:54:42"	"TodayCleveland '
89840384	"march_for_trump"	"2016-10-02 02:35:35"	"NickTomaWBRE Hi,
73691904	"puredavie"	"2016-12-21 14:35:32"	"hat. Is. A. Reso
35774720	"daileyjadon"	"2016-11-21 11:04:00"	"ifetime movie
28585728	"evagreen69"	"2017-01-20 02:16:36"	"T @Conservatexia
82891776	"cassieweltch"	"2016-11-07 16:16:18"	"T @HillaryClinto
79158784	"_nickluna_"	"2016-09-19 13:20:08"	"T @leonpui_: Hil

47
10
15
15
15
15

"n.name"	"count"
"Trump "	8029
"CNN "	1263
"GOP "	914
"FBI "	762
"TRUMP "	675
"Clinton "	672
"ISIS "	541
"MAGA "	479
"YouTube "	411
"Twitter "	403
"wikileaks "	377
"Obama "	352
"HRC "	340
"America "	335
"Wikileaks "	321

Relacyjne bazy danych

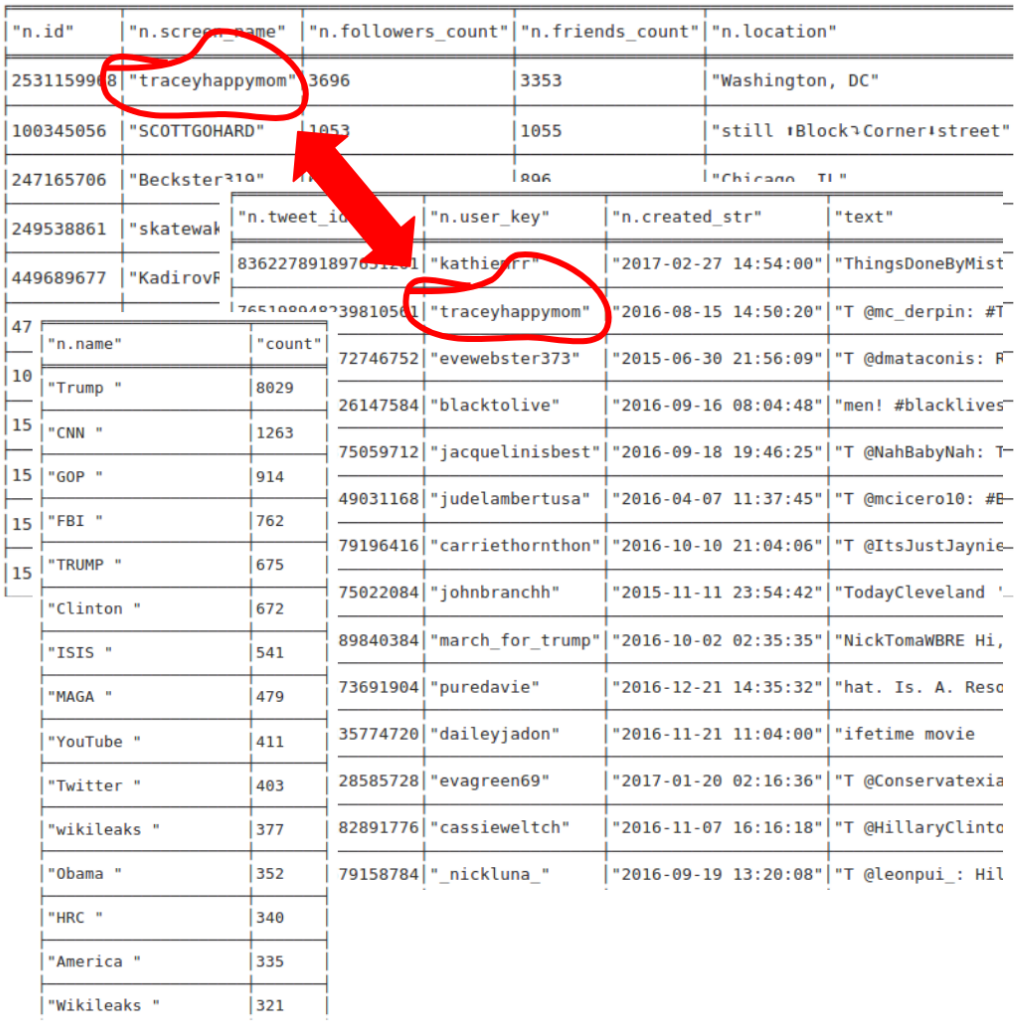


"n.id"	"n.screen_name"	"n.followers_count"	"n.friends_count"	"n.location"
253115990[illegible]	"traceyhappymom"	3696	3353	"Washington, DC"
100345056	"SCOTTGOHARD"	1053	1055	"still ıBlockıCornerıstreet"
247165706	"Beckster319"	[illegible]	896	"Chicago, IL"
249538861	"skatewak[illegible]			
449689677	"KadirovF[illegible]			

"n.tweet_id"	"n.user_key"	"n.created_str"	"text"
836227891897651201	"kathiexr"	"2017-02-27 14:54:00"	"ThingsDoneByMist
765198948398105601	"traceyhappymom"	"2016-08-15 14:50:20"	"T @mc_derpin: #T
72746752	"evewebster373"	"2015-06-30 21:56:09"	"T @dmataconis: R
26147584	"blacktolive"	"2016-09-16 08:04:48"	"men! #blacklives
75059712	"jacquelinisbest"	"2016-09-18 19:46:25"	"T @NahBabyNah: T
49031168	"judelambertusa"	"2016-04-07 11:37:45"	"T @mcicero10: #B
79196416	"carriethornthon"	"2016-10-10 21:04:06"	"T @ItsJustJaynie
75022084	"johnbranchh"	"2015-11-11 23:54:42"	"TodayCleveland '
89840384	"march_for_trump"	"2016-10-02 02:35:35"	"NickTomaWBRE Hi,
73691904	"puredavie"	"2016-12-21 14:35:32"	"hat. Is. A. Reso
35774720	"daileyjadon"	"2016-11-21 11:04:00"	"ifetime movie
28585728	"evagreen69"	"2017-01-20 02:16:36"	"T @Conservatexia
82891776	"cassieweltch"	"2016-11-07 16:16:18"	"T @HillaryClinto
79158784	"_nickluna_"	"2016-09-19 13:20:08"	"T @leonpui_: Hil

47	"n.name"	"count"
10	"Trump "	8029
15	"CNN "	1263
15	"GOP "	914
15	"FBI "	762
15	"TRUMP "	675
	"Clinton "	672
	"ISIS "	541
	"MAGA "	479
	"YouTube "	411
	"Twitter "	403
	"wikileaks "	377
	"Obama "	352
	"HRC "	340
	"America "	335
	"Wikileaks "	321

Relacyjne bazy danych



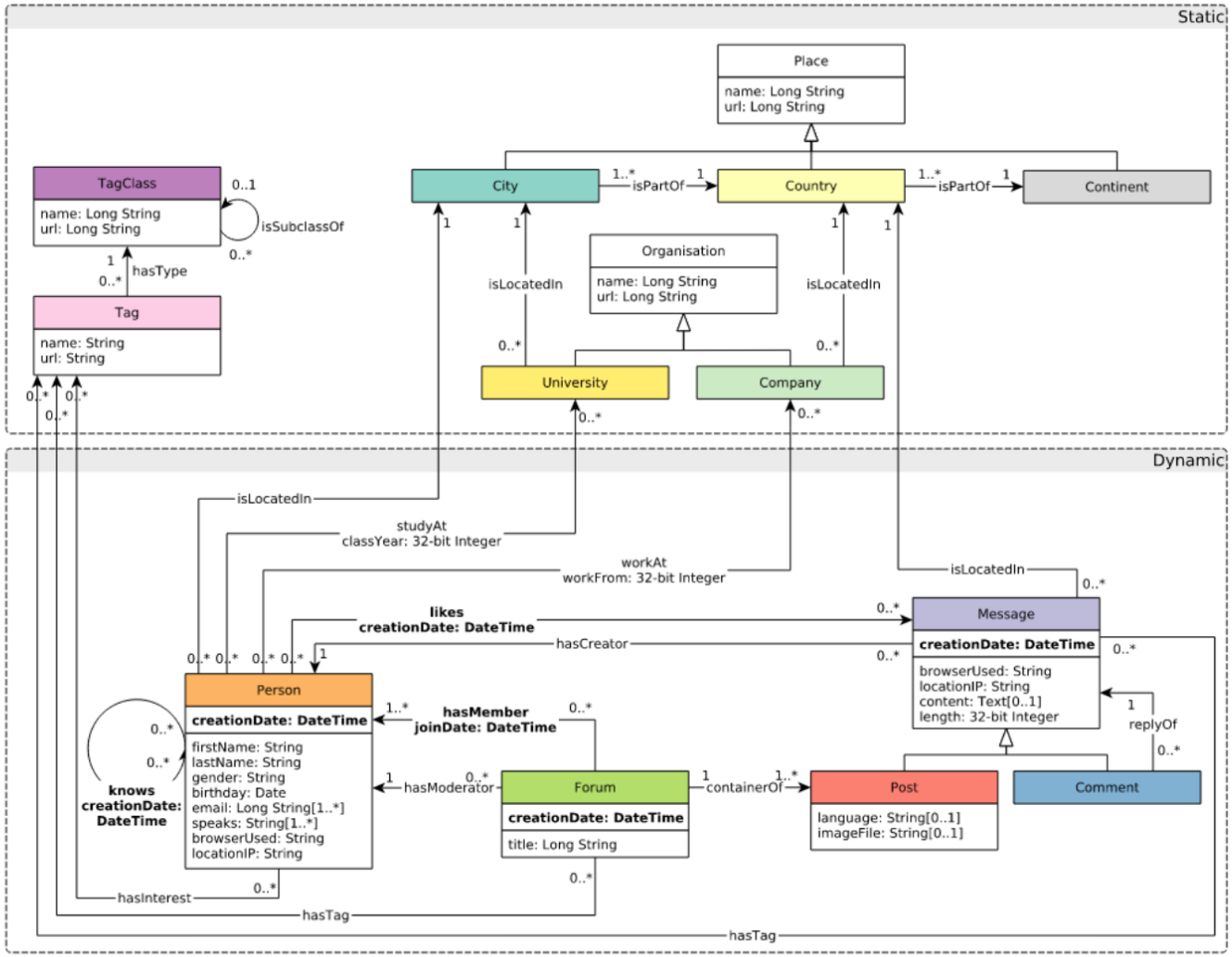
"n.id"	"n.screen_name"	"n.followers_count"	"n.friends_count"	"n.location"
253115990[illegible]8	"traceyhappymom"	3696	3353	"Washington, DC"
100345056	"SCOTTGOHARD"	1053	1055	"still ıBlockıCornerıstreet"
247165706	"Beckster319"	[illegible]	896	"Chicago, IL"
249538861	"skatewak			
449689677	"KadirovF			

"n.tweet_id"	"n.user_key"	"n.created_str"	"text"
836227891897651201	"kathiemr"	"2017-02-27 14:54:00"	"ThingsDoneByMist
765198948239810561	"traceyhappymom"	"2016-08-15 14:50:20"	"T @mc_derpin: #T
72746752	"evewebster373"	"2015-06-30 21:56:09"	"T @dmataconis: R
26147584	"blacktolive"	"2016-09-16 08:04:48"	"men! #blacklives
75059712	"jacquelinisbest"	"2016-09-18 19:46:25"	"T @NahBabyNah: T
49031168	"judelambertusa"	"2016-04-07 11:37:45"	"T @mcicero10: #B
79196416	"carriethornthon"	"2016-10-10 21:04:06"	"T @ItsJustJaynie
75022084	"johnbranchh"	"2015-11-11 23:54:42"	"TodayCleveland '
89840384	"march_for_trump"	"2016-10-02 02:35:35"	"NickTomaWBRE Hi,
73691904	"puredavie"	"2016-12-21 14:35:32"	"hat. Is. A. Reso
35774720	"daileyjadon"	"2016-11-21 11:04:00"	"ifetime movie
28585728	"evagreen69"	"2017-01-20 02:16:36"	"T @Conservatexia
82891776	"cassieweltch"	"2016-11-07 16:16:18"	"T @HillaryClinto
79158784	"_nickluna_"	"2016-09-19 13:20:08"	"T @leonpui_: Hil

47	
"n.name"	"count"
"Trump "	8029
"CNN "	1263
"GOP "	914
"FBI "	762
"TRUMP "	675
"Clinton "	672
"ISIS "	541
"MAGA "	479
"YouTube "	411
"Twitter "	403
"wikileaks "	377
"Obama "	352
"HRC "	340
"America "	335
"Wikileaks "	321

W jakich godzinach tweetują osoby z *Washington, DC*?

Relacyjne bazy danych



LDBC Benchmark

Relacyjne bazy danych

- prosty model, łatwy w użyciu,

Relacyjne bazy danych

- prosty model, łatwy w użyciu,
- dane tabelaryczne, nacisk na wartości danych (a nie np. topologię połączeń)

Relacyjne bazy danych

- prosty model, łatwy w użyciu,
- dane tabelaryczne, nacisk na wartości danych (a nie np. topologię połączeń)
- analiza danych, przegrupowania i agregacje,

Relacyjne bazy danych

- prosty model, łatwy w użyciu,
- dane tabelaryczne, nacisk na wartości danych (a nie np. topologię połączeń)
- analiza danych, przegrupowania i agregacje,
- normalizacja - logiczna organizacja danych, brak redundacji i anomalii,

Relacyjne bazy danych

- prosty model, łatwy w użyciu,
- dane tabelaryczne, nacisk na wartości danych (a nie np. topologię połączeń)
- analiza danych, przegrupowania i agregacje,
- normalizacja - logiczna organizacja danych, brak redundacji i anomalii,

- nie zawsze z góry wiadomo jaka powinna być struktura bazy, jakie dane dokładnie? takie same dla każdego obiektu?

Relacyjne bazy danych

- prosty model, łatwy w użyciu,
- dane tabelaryczne, nacisk na wartości danych (a nie np. topologię połączeń)
- analiza danych, przegrupowania i agregacje,
- normalizacja - logiczna organizacja danych, brak redundacji i anomalii,

- nie zawsze z góry wiadomo jaka powinna być struktura bazy, jakie dane dokładnie? takie same dla każdego obiektu?
- normalizacja $\rightarrow$ wiele tabel, rozdrobnienie, dużo złączeń,

Relacyjne bazy danych

- prosty model, łatwy w użyciu,
- dane tabelaryczne, nacisk na wartości danych (a nie np. topologię połączeń)
- analiza danych, przegrupowania i agregacje,
- normalizacja - logiczna organizacja danych, brak redundacji i anomalii,

- nie zawsze z góry wiadomo jaka powinna być struktura bazy, jakie dane dokładnie? takie same dla każdego obiektu?
- normalizacja $\rightarrow$ wiele tabel, rozdrobnienie, dużo złączeń,
- podejście zwinne - częste zmiany schematu?

Relacyjne bazy danych

- prosty model, łatwy w użyciu,
- dane tabelaryczne, nacisk na wartości danych (a nie np. topologię połączeń)
- analiza danych, przegrupowania i agregacje,
- normalizacja - logiczna organizacja danych, brak redundacji i anomalii,

- nie zawsze z góry wiadomo jaka powinna być struktura bazy, jakie dane dokładnie? takie same dla każdego obiektu?
- normalizacja $\rightarrow$ wiele tabel, rozdrobnienie, dużo złączeń,
- podejście zwinne - częste zmiany schematu?
- migracje schematu gdy masz duże dane - kosztowne,

Relacyjne bazy danych

- prosty model, łatwy w użyciu,
- dane tabelaryczne, nacisk na wartości danych (a nie np. topologię połączeń)
- analiza danych, przegrupowania i agregacje,
- normalizacja - logiczna organizacja danych, brak redundacji i anomalii,

- nie zawsze z góry wiadomo jaka powinna być struktura bazy, jakie dane dokładnie? takie same dla każdego obiektu?
- normalizacja $\rightarrow$ wiele tabel, rozdrobnienie, dużo złączeń,
- podejście zwinne - częste zmiany schematu?
- migracje schematu gdy masz duże dane - kosztowne,
- skalowanie poziome (scale-up vs. scale-out), rozpraszanie, chmury, spójność & dostępność.

NoSQL: key-values stores, document stores

- brak schematu / elastyczny schemat,

NoSQL: key-values stores, document stores

- brak schematu / elastyczny schemat,
- brak normalizacji $\rightarrow$ redundancja, anomalie,

NoSQL: key-values stores, document stores

- brak schematu / elastyczny schemat,
- brak normalizacji $\rightarrow$ redundancja, anomalie,
- duża wydajność/skalowalność kosztem organizacji danych wg pojedynczego klucza (ale są indeksy),

NoSQL: key-values stores, document stores

- brak schematu / elastyczny schemat,
- brak normalizacji $\rightarrow$ redundancja, anomalie,
- duża wydajność/skalowalność kosztem organizacji danych wg pojedynczego klucza (ale są indeksy),
- brak łatwej możliwości modelowania związków pomiędzy danymi (dodawanie `id` jednego dokumentu jako wartość w drugim...),

NoSQL: key-values stores, document stores

- brak schematu / elastyczny schemat,
- brak normalizacji $\rightarrow$ redundancja, anomalie,
- duża wydajność/skalowalność kosztem organizacji danych wg pojedynczego klucza (ale są indeksy),
- brak łatwej możliwości modelowania związków pomiędzy danymi (dodawanie `id` jednego dokumentu jako wartość w drugim...),
- przegrupowanie danych, joiny $\rightarrow$ obliczenia po stronie aplikacji,

NoSQL: key-values stores, document stores

- brak schematu / elastyczny schemat,
- brak normalizacji $\rightarrow$ redundancja, anomalie,
- duża wydajność/skalowalność kosztem organizacji danych wg pojedynczego klucza (ale są indeksy),
- brak łatwej możliwości modelowania związków pomiędzy danymi (dodawanie `id` jednego dokumentu jako wartość w drugim...),
- przegrupowanie danych, joiny $\rightarrow$ obliczenia po stronie aplikacji,
- przechowywanie już zjoinowanych danych (join at write-time),

NoSQL: key-values stores, document stores

- brak schematu / elastyczny schemat,
- brak normalizacji $\rightarrow$ redundancja, anomalie,
- duża wydajność/skalowalność kosztem organizacji danych wg pojedynczego klucza (ale są indeksy),
- brak łatwej możliwości modelowania związków pomiędzy danymi (dodawanie `id` jednego dokumentu jako wartość w drugim...),
- przegrupowanie danych, joiny $\rightarrow$ obliczenia po stronie aplikacji,
- przechowywanie już zjoinowanych danych (join at write-time), szybkość odczytu!

NoSQL: key-values stores, document stores

- brak schematu / elastyczny schemat,
- brak normalizacji $\rightarrow$ redundancja, anomalie,
- duża wydajność/skalowalność kosztem organizacji danych wg pojedynczego klucza (ale są indeksy),
- brak łatwej możliwości modelowania związków pomiędzy danymi (dodawanie `id` jednego dokumentu jako wartość w drugim...),
- przegrupowanie danych, joiny $\rightarrow$ obliczenia po stronie aplikacji,
- przechowywanie już zjoinowanych danych (join at write-time), szybkość odczytu!
- typowe zastosowania: profile użytkownika (web, gry, ...), koszyk zakupowy, rekomendacje, ogłoszenia, cache.

NoSQL: key-values stores, document stores

- brak schematu / elastyczny schemat,
- brak normalizacji $\rightarrow$ redundancja, anomalie,
- duża wydajność/skalowalność kosztem organizacji danych wg pojedynczego klucza (ale są indeksy),
- brak łatwej możliwości modelowania związków pomiędzy danymi (dodawanie `id` jednego dokumentu jako wartość w drugim...),
- przegrupowanie danych, joiny $\rightarrow$ obliczenia po stronie aplikacji,
- przechowywanie już zjoinowanych danych (join at write-time), szybkość odczytu!
- typowe zastosowania: profile użytkownika (web, gry, ...), koszyk zakupowy, rekomendacje, ogłoszenia, cache.

NoSQL: key-values stores, document stores

- brak schematu / elastyczny schemat,
- brak normalizacji → redundancja, anomalie,
- duża wydajność/skalowalność kosztem organizacji danych wg pojedynczego klucza (ale są indeksy),
- brak łatwej możliwości modelowania związków pomiędzy danymi (dodawanie `id` jednego dokumentu jako wartość w drugim...),
- przegrupowanie danych, joiny → obliczenia po stronie aplikacji,
- przechowywanie już zjoinowanych danych (join at write-time), szybkość odczytu!
- typowe zastosowania: profile użytkownika (web, gry, ...), koszyk zakupowy, rekomendacje, ogłoszenia, cache.

```
{   "id": "PW1",
    "name": "Piotr Wieczorek",
    "company": {
          "name": "II UWr",
          "otherdata": { "here": "a lot of (pre-joined) data on II UWr" },
    "latitude": 51.110922126668 58,
    "longitude": 17.05313385578201,
    "expertise": [
        "graph databases",
        "PostgreSQL",
        "MongoDB"
    ]
}
```

https://www.json.org/json-en.html

Bazy grafowe

- organizacja danych wspiera chodzenie po krawędziach,

Bazy grafowe

- organizacja danych wspiera chodzenie po krawędziach,
- o wiele prostszy, naturalny model danych,

Bazy grafowe

- organizacja danych wspiera chodzenie po krawędziach,
- o wiele prostszy, naturalny model danych,
- property graphs,

Bazy grafowe

- organizacja danych wspiera chodzenie po krawędziach,
- o wiele prostszy, naturalny model danych,
- property graphs,
- zapytania ścieżkowe (wyrażenia regularne),

Bazy grafowe

- organizacja danych wspiera chodzenie po krawędziach,
- o wiele prostszy, naturalny model danych,
- property graphs,
- zapytania ścieżkowe (wyrażenia regularne),
- typowe zastosowania: wyszukiwanie połączeń pomiędzy wierzchołkami, bazy wiedzy, dane o sieciach, rekomendacje, fraudy, paczki, farmacja, śledztwa dziennikarskie (np. Panama Papers), centralność (page-rank) *znajdź bossa mafii*, itp.

Bazy grafowe

- organizacja danych wspiera chodzenie po krawędziach,
- o wiele prostszy, naturalny model danych,
- property graphs,
- zapytania ścieżkowe (wyrażenia regularne),
- typowe zastosowania: wyszukiwanie połączeń pomiędzy wierzchołkami, bazy wiedzy, dane o sieciach, rekomendacje, fraudy, paczki, farmacja, śledztwa dziennikarskie (np. Panama Papers), centralność (page-rank) *znajdź bossa mafii*, itp.

Bazy grafowe

- organizacja danych wspiera chodzenie po krawędziach,
- o wiele prostszy, naturalny model danych,
- property graphs,
- zapytania ścieżkowe (wyrażenia regularne),
- typowe zastosowania: wyszukiwanie połączeń pomiędzy wierzchołkami, bazy wiedzy, dane o sieciach, rekomendacje, fraudy, paczki, farmacja, śledztwa dziennikarskie (np. Panama Papers), centralność (page-rank) *znajdź bossa mafii*, itp.

```
MATCH (p:Product {productName:"Chocolade"})<-[:PRODUCT]
                            -(:Order)<-[:PURCHASED]-(c:Customer)
RETURN distinct c.companyName;
```

PostGIS

- **How far is New York from Seattle? What are the units of the answer?**

> **Note**
>
> New York = `POINT(-74.0064 40.7142)` and Seattle = `POINT(-122.3331 47.6097)`

```
SELECT ST_Distance(
  'POINT(-74.0064 40.7142)'::geography,
  'POINT(-122.3331 47.6097)'::geography
  );
```

```
3875538.57141352
```

- **What is the total length of all streets in New York, calculated on the spheroid?**

```
SELECT Sum(
  ST_Length(Geography(
    ST_Transform(geom,4326)
  )))
FROM nyc_streets;
```

```
10421999.666
```

Bazy wektorowe (np. Kdb, Pinecone)

Vector Search engines provide the ability for developers to store vectors structured around certain algorithms (i.e. KNN), and an engine to compute similar vectors (like cosine distance) to determine which vectors are related.
(`http://vectorsearch.dev/`)

Plan wykładu

Będzie o:

1. **Model relacyjny teoretycznie:** elementy składowe modelu, języki zapytań, zapytania koniunkcyjne, rekursja, postaci normalne (BCNF, 3NF, 4NF).
2. **Model relacyjny praktycznie:** zapytania SQL, projektowanie baz danych oraz diagramy E-R i UML, język definicji danych SQL.
3. **Systemy zarządzania relacyjnymi bazami danych:** przetwarzanie zapytań, transakcje i wielodostęp, bezpieczeństwo danych.
4. **Inne modele** - dokumentowe i grafowe bazy danych (tylko troszkę).

Ćwiczenia i pracownia

Będziemy się uczyć:

1. O modelu relacyjnym (algebra relacji i rachunki relacyjne, postaci normalne);

Ćwiczenia i pracownia

Będziemy się uczyć:

1. O modelu relacyjnym (algebra relacji i rachunki relacyjne, postaci normalne);
2. Jak korzystać z gotowej bazy danych — wyszukiwać w niej informacje, odpowiedzi na interesujące nas pytania (język SQL);

Ćwiczenia i pracownia

Będziemy się uczyć:

1. O modelu relacyjnym (algebra relacji i rachunki relacyjne, postaci normalne);
2. Jak korzystać z gotowej bazy danych — wyszukiwać w niej informacje, odpowiedzi na interesujące nas pytania (język SQL);
3. Jak konstruować poprawne bazy danych dla zagadnień rzeczywistych — projektować bazy (modelować) i na podstawie projektów definiować elementy baz danych;

Ćwiczenia i pracownia

Będziemy się uczyć:

1. O modelu relacyjnym (algebra relacji i rachunki relacyjne, postaci normalne);
2. Jak korzystać z gotowej bazy danych — wyszukiwać w niej informacje, odpowiedzi na interesujące nas pytania (język SQL);
3. Jak konstruować poprawne bazy danych dla zagadnień rzeczywistych — projektować bazy (modelować) i na podstawie projektów definiować elementy baz danych;
4. Jak tworzyć aplikacje korzystające z bazy danych.

Ćwiczenia i pracownia

Będziemy się uczyć:

1. O modelu relacyjnym (algebra relacji i rachunki relacyjne, postaci normalne);
2. Jak korzystać z gotowej bazy danych — wyszukiwać w niej informacje, odpowiedzi na interesujące nas pytania (język SQL);
3. Jak konstruować poprawne bazy danych dla zagadnień rzeczywistych — projektować bazy (modelować) i na podstawie projektów definiować elementy baz danych;
4. Jak tworzyć aplikacje korzystające z bazy danych.

Ćwiczenia i pracownia

Będziemy się uczyć:

1. O modelu relacyjnym (algebra relacji i rachunki relacyjne, postaci normalne);
2. Jak korzystać z gotowej bazy danych — wyszukiwać w niej informacje, odpowiedzi na interesujące nas pytania (język SQL);
3. Jak konstruować poprawne bazy danych dla zagadnień rzeczywistych — projektować bazy (modelować) i na podstawie projektów definiować elementy baz danych;
4. Jak tworzyć aplikacje korzystające z bazy danych.

Materiały i informacje: `https://skos.ii.uni.wroc.pl/course/view.php?id=647`
— kurs Bazy Danych 2024.

Plan

Elementy modelu

Elementy modelu

Relacja (tabela) — jedyna struktura dla danych w modelu; ma ustaloną liczbę kolumn, w które można wpisywać wartości ustalonego typu i dowolną liczbę wierszy.

Elementy modelu

Relacja (tabela) — jedyna struktura dla danych w modelu; ma ustaloną liczbę kolumn, w które można wpisywać wartości ustalonego typu i dowolną liczbę wierszy.

Więzy (warunki poprawności, warunki spójności) — dane wpisywane do tabel muszą spełniać zdefiniowane warunki: typ danych, zakres,...

Elementy modelu

Relacja (tabela) — jedyna struktura dla danych w modelu; ma ustaloną liczbę kolumn, w które można wpisywać wartości ustalonego typu i dowolną liczbę wierszy.

Więzy (warunki poprawności, warunki spójności) — dane wpisywane do tabel muszą spełniać zdefiniowane warunki: typ danych, zakres,...

Baza danych — zbiór tabel z danymi spełniającymi nałożone na nie więzy.

Elementy modelu

Relacja (tabela) — jedyna struktura dla danych w modelu; ma ustaloną liczbę kolumn, w które można wpisywać wartości ustalonego typu i dowolną liczbę wierszy.

Więzy (warunki poprawności, warunki spójności) — dane wpisywane do tabel muszą spełniać zdefiniowane warunki: typ danych, zakres,...

Baza danych — zbiór tabel z danymi spełniającymi nałożone na nie więzy.

Języki zapytań (*query language*) — języki pozwalające wyszukać w relacjach określoną informację.

Plan

Relacja, czyli tabela

Osoba

Nazwisko : `varchar(20)`	PESEL : `char(11)`	dataUr : `date`
...		
Abacki	80121304455	‘20-02-1980’
...		

Mieszkanie

PESEL : `char(11)`	Adres : `varchar(50)`	Metraż : `real`
...		
80121304455	Ełk, Kwiatowa 100	60,2
80121304455	Poznań, Szeroka 10/2	30,2
NULL	Ełk, Kwiatowa 102	64,2
...		

Elementy relacji

- *Atrybut* — nazwa kolumny;
- *Dziedzina* — typ danych;
- *Krotność (arność)* — liczba atrybutów;
- *Krotka (wiersz)* — element relacji;
- *Atrybuty krotki* — *Osoba*[3] lub *Mieszkanie.Adres*;
- *Schemat relacji* — nazwa relacji, nazwy i typy kolumn;
- *Stan relacji* to zawarte w niej krotki.

Relacja, czyli tabela

Osoba

Nazwisko : `varchar(20)`	PESEL : `char(11)`	dataUr : `date`
...		
Abacki	80121304455	‘20-02-1980’
...		

Mieszkanie

PESEL : `char(11)`	Adres : `varchar(50)`	Metraż : `real`
...		
80121304455	Ełk, Kwiatowa 100	60,2
80121304455	Poznań, Szeroka 10/2	30,2
NULL	Ełk, Kwiatowa 102	64,2
...		

Elementy relacji

- *Atrybut* — nazwa kolumny;
- *Dziedzina* — typ danych;
- *Krotność (arność)* — liczba atrybutów;
- *Krotka (wiersz)* — element relacji;
- *Atrybuty krotki* — *Osoba*[3] lub *Mieszkanie.Adres*;
- *Schemat relacji* — nazwa relacji, nazwy i typy kolumn;
- *Stan relacji* to zawarte w niej krotki.

Relacja, czyli tabela

Osoba

Nazwisko : `varchar(20)`	PESEL : `char(11)`	dataUr : `date`
...		
Abacki	80121304455	‘20-02-1980’
...		

Mieszkanie

PESEL : `char(11)`	Adres : `varchar(50)`	Metraż : `real`
...		
80121304455	Ełk, Kwiatowa 100	60,2
80121304455	Poznań, Szeroka 10/2	30,2
NULL	Ełk, Kwiatowa 102	64,2
...		

Elementy relacji

- *Atrybut* — nazwa kolumny;
- *Dziedzina* — typ danych;
- *Krotność (arność)* — liczba atrybutów;
- *Krotka (wiersz)* — element relacji;
- *Atrybuty krotki* — *Osoba*[3] lub *Mieszkanie.Adres*;
- *Schemat relacji* — nazwa relacji, nazwy i typy kolumn;
- *Stan relacji* to zawarte w niej krotki.

Relacja, czyli tabela

Osoba

Nazwisko : `varchar(20)`	PESEL : `char(11)`	dataUr : `date`
...		
Abacki	80121304455	‘20-02-1980’
...		

Mieszkanie

PESEL : `char(11)`	Adres : `varchar(50)`	Metraż : `real`
...		
80121304455	Ełk, Kwiatowa 100	60,2
80121304455	Poznań, Szeroka 10/2	30,2
NULL	Ełk, Kwiatowa 102	64,2
...		

Elementy relacji

- *Atrybut* — nazwa kolumny;
- *Dziedzina* — typ danych;
- *Krotność (arność)* — liczba atrybutów;
- *Krotka (wiersz)* — element relacji;
- *Atrybuty krotki* — *Osoba*[3] lub *Mieszkanie.Adres*;
- *Schemat relacji* — nazwa relacji, nazwy i typy kolumn;
- *Stan relacji* to zawarte w niej krotki.

Relacja, czyli tabela

Osoba

Nazwisko : `varchar(20)`	PESEL : `char(11)`	dataUr : `date`
...		
Abacki	80121304455	‘20-02-1980’
...		

Mieszkanie

PESEL : `char(11)`	Adres : `varchar(50)`	Metraż : `real`
...		
80121304455	Ełk, Kwiatowa 100	60,2
80121304455	Poznań, Szeroka 10/2	30,2
NULL	Ełk, Kwiatowa 102	64,2
...		

Elementy relacji

- *Atrybut* — nazwa kolumny;
- *Dziedzina* — typ danych;
- *Krotność (arność)* — liczba atrybutów;
- *Krotka (wiersz)* — element relacji;
- *Atrybuty krotki* — *Osoba*[3] lub *Mieszkanie.Adres*;
- *Schemat relacji* — nazwa relacji, nazwy i typy kolumn;
- *Stan relacji* to zawarte w niej krotki.

Relacja, czyli tabela

Osoba

Nazwisko : `varchar(20)`	PESEL : `char(11)`	dataUr : `date`
...		
Abacki	80121304455	'20-02-1980'
...		

Mieszkanie

PESEL : `char(11)`	Adres : `varchar(50)`	Metraż : `real`
...		
80121304455	Ełk, Kwiatowa 100	60,2
80121304455	Poznań, Szeroka 10/2	30,2
NULL	Ełk, Kwiatowa 102	64,2
...		

Elementy relacji

- *Atrybut* — nazwa kolumny;
- *Dziedzina* — typ danych;
- *Krotność (arność)* — liczba atrybutów;
- *Krotka (wiersz)* — element relacji;
- *Atrybuty krotki — Osoba*[3] lub *Mieszkanie.Adres*;
- *Schemat relacji* — nazwa relacji, nazwy i typy kolumn;
- *Stan relacji* to zawarte w niej krotki.

Relacja, czyli tabela

Osoba

Nazwisko : `varchar(20)`	PESEL : `char(11)`	dataUr : `date`
...		
Abacki	80121304455	‘20-02-1980’
...		

Mieszkanie

PESEL : `char(11)`	Adres : `varchar(50)`	Metraż : `real`
...		
80121304455	Ełk, Kwiatowa 100	60,2
80121304455	Poznań, Szeroka 10/2	30,2
NULL	Ełk, Kwiatowa 102	64,2
...		

Elementy relacji

- *Atrybut* — nazwa kolumny;
- *Dziedzina* — typ danych;
- *Krotność (arność)* — liczba atrybutów;
- *Krotka (wiersz)* — element relacji;
- *Atrybuty krotki* — *Osoba*[3] lub *Mieszkanie.Adres*;
- *Schemat relacji* — nazwa relacji, nazwy i typy kolumn;
- *Stan relacji* to zawarte w niej krotki.

Relacja, czyli tabela

Osoba

Nazwisko : `varchar(20)`	PESEL : `char(11)`	dataUr : `date`
...		
Abacki	80121304455	‘20-02-1980’
...		

Mieszkanie

PESEL : `char(11)`	Adres : `varchar(50)`	Metraż : `real`
...		
80121304455	Ełk, Kwiatowa 100	60,2
80121304455	Poznań, Szeroka 10/2	30,2
NULL	Ełk, Kwiatowa 102	64,2
...		

Elementy relacji

- *Atrybut* — nazwa kolumny;
- *Dziedzina* — typ danych;
- *Krotność (arność)* — liczba atrybutów;
- *Krotka (wiersz)* — element relacji;
- *Atrybuty krotki* — *Osoba*[3] lub *Mieszkanie.Adres*;
- *Schemat relacji* — nazwa relacji, nazwy i typy kolumn;
- *Stan relacji* to zawarte w niej krotki.

Notacja matematyczna

Dla atrybutów $A_1, \ldots, A_k$ i związanych z nimi dziedzin $D_1, \ldots, D_k$ relacja R ma:

schemat $R = A_1 \ldots A_k$ lub $R(A_1, \ldots, A_k)$,

arność k,

stan $r \subseteq D_1 \times \cdots \times D_k$,

krotki $(v_1, v_2, \ldots, v_k) \in r$.

Relacyjna baza danych (schemat i stan) to zbiór relacji o różnych nazwach.

Notacja matematyczna

Dla atrybutów $A_1, \ldots, A_k$ i związanych z nimi dziedzin $D_1, \ldots, D_k$ relacja R ma:

schemat $R = A_1 \ldots A_k$ lub $R(A_1, \ldots, A_k)$,

arność k,

stan $r \subseteq D_1 \times \cdots \times D_k$,

krotki $(v_1, v_2, \ldots, v_k) \in r$.

Relacyjna baza danych (schemat i stan) to zbiór relacji o różnych nazwach.

W przykładzie:

- Osoba(Nazwisko,PESEL,dataUr),
- Mieszkanie(PESEL,Adres,Metraż)

Plan

Klucze

Klucze

Klucz relacji

Podzbiór atrybutów relacji, których wartości zawsze pozwalają jednoznacznie zidentyfikować krotkę relacji. Oznacza, to że nie dopuszczamy, by w danych znalazły się dwie różne krotki o jednakowych wartościach klucza. Relacja może mieć kilka kluczy:
`Student(indeks,PESEL,Nazwisko,...)`

Klucze

Klucz relacji

Podzbiór atrybutów relacji, których wartości zawsze pozwalają jednoznacznie zidentyfikować krotkę relacji. Oznacza, to że nie dopuszczamy, by w danych znalazły się dwie różne krotki o jednakowych wartościach klucza. Relacja może mieć kilka kluczy:
`Student(indeks,PESEL,Nazwisko,...)`

Klucz główny

Jeden z kluczy relacji. Zazwyczaj wybieramy ten, według którego najczęściej będziemy wyszukiwać dane z relacji. Pozostałe klucze nazywamy *kandydującymi* lub *alternatywnymi*. Na przykład `indeks` może być kluczem głównym relacji `Student`, a `PESEL` — kluczem alternatywnym.

Klucze

Klucz relacji

Podzbiór atrybutów relacji, których wartości zawsze pozwalają jednoznacznie zidentyfikować krotkę relacji. Oznacza, to że nie dopuszczamy, by w danych znalazły się dwie różne krotki o jednakowych wartościach klucza. Relacja może mieć kilka kluczy:
`Student(indeks,PESEL,Nazwisko,...)`

Klucz główny

Jeden z kluczy relacji. Zazwyczaj wybieramy ten, według którego najczęściej będziemy wyszukiwać dane z relacji. Pozostałe klucze nazywamy *kandydującymi* lub *alternatywnymi*. Na przykład `indeks` może być kluczem głównym relacji `Student`, a `PESEL` — kluczem alternatywnym.

Klucz z wielu atrybutów

Stosujemy takie rozwiązanie, gdy jeden atrybut nie wystarcza do zidentyfikowania krotki. Na przykład w relacji `Zaliczenie(indeks,kod_przedmiotu,ocena,data)`.

Klucz obcy

Dane w bazie muszą często zostać rozmieszczone w różnych relacjach, pomimo że się ze soba wiążą. Do połączenia danych z różnych relacji służą **klucze obce**.

Zaliczenie:

indeks	kod_przedm	ocena
123456	BD2011	5.0
123456	SK2011	4.5
654321	BD2011	3.5

Student:

indeks	PESEL	nazwisko
123456	AB123456	Abacka
654321	CD345678	Babacka
987654	DE534343	Cabacka

Klucz obcy

Dane w bazie muszą często zostać rozmieszczone w różnych relacjach, pomimo że się ze soba wiążą. Do połączenia danych z różnych relacji służą **klucze obce**.

Zaliczenie:

indeks	kod_przedm	ocena
123456	BD2011	5.0
123456	SK2011	4.5
654321	BD2011	3.5

Student:

indeks	PESEL	nazwisko
123456	AB123456	Abacka
654321	CD345678	Babacka
987654	DE534343	Cabacka

- Zamieszczony w relacji `Zaliczenie` atrybut `indeks` służy do zidentyfikowania osoby z relacji `Student`.

Klucz obcy

Dane w bazie muszą często zostać rozmieszczone w różnych relacjach, pomimo że się ze soba wiążą. Do połączenia danych z różnych relacji służą **klucze obce**.

Zaliczenie:

indeks	kod_przedm	ocena
123456	BD2011	5.0
123456	SK2011	4.5
654321	BD2011	3.5

Student:

indeks	PESEL	nazwisko
123456	AB123456	Abacka
654321	CD345678	Babacka
987654	DE534343	Cabacka

- Zamieszczony w relacji `Zaliczenie` atrybut `indeks` służy do zidentyfikowania osoby z relacji `Student`.
- W relacji `Student` atrybut `indeks` jest kluczem.

Klucz obcy

Dane w bazie muszą często zostać rozmieszczone w różnych relacjach, pomimo że się ze soba wiążą. Do połączenia danych z różnych relacji służą **klucze obce**.

Zaliczenie:

indeks	kod_przedm	ocena
123456	BD2011	5.0
123456	SK2011	4.5
654321	BD2011	3.5

Student:

indeks	PESEL	nazwisko
123456	AB123456	Abacka
654321	CD345678	Babacka
987654	DE534343	Cabacka

- Zamieszczony w relacji `Zaliczenie` atrybut `indeks` służy do zidentyfikowania osoby z relacji `Student`.
- W relacji `Student` atrybut `indeks` jest kluczem.
- W relacji `Zaliczenie` atrybut `indeks` może powtarzać się lub być pusty.

Klucz obcy

Dane w bazie muszą często zostać rozmieszczone w różnych relacjach, pomimo że się ze soba wiążą. Do połączenia danych z różnych relacji służą **klucze obce**.

Zaliczenie:

indeks	kod_przedm	ocena
123456	BD2011	5.0
123456	SK2011	4.5
654321	BD2011	3.5

Student:

indeks	PESEL	nazwisko
123456	AB123456	Abacka
654321	CD345678	Babacka
987654	DE534343	Cabacka

- Zamieszczony w relacji `Zaliczenie` atrybut `indeks` służy do zidentyfikowania osoby z relacji `Student`.
- W relacji `Student` atrybut `indeks` jest kluczem.
- W relacji `Zaliczenie` atrybut `indeks` może powtarzać się lub być pusty.
- Jeśli indeks jest użyty w relacji `Zaliczenie`, to w relacji `Student` powinna występować osoba o tym indeksie (integralność referencyjna).

Klucz obcy

Dane w bazie muszą często zostać rozmieszczone w różnych relacjach, pomimo że się ze soba wiążą. Do połączenia danych z różnych relacji służą **klucze obce**.

Zaliczenie:

indeks	kod_przedm	ocena
123456	BD2011	5.0
123456	SK2011	4.5
654321	BD2011	3.5
999999	BD2012	2.0

Student:

indeks	PESEL	nazwisko
123456	AB123456	Abacka
654321	CD345678	Babacka
987654	DE534343	Cabacka

- Zamieszczony w relacji `Zaliczenie` atrybut `indeks` służy do zidentyfikowania osoby z relacji `Student`.
- W relacji `Student` atrybut `indeks` jest kluczem.
- W relacji `Zaliczenie` atrybut `indeks` może powtarzać się lub być pusty.
- Jeśli indeks jest użyty w relacji `Zaliczenie`, to w relacji `Student` powinna występować osoba o tym indeksie (integralność referencyjna).

Klucz obcy

Dane w bazie muszą często zostać rozmieszczone w różnych relacjach, pomimo że się ze soba wiążą. Do połączenia danych z różnych relacji służą **klucze obce**.

Zaliczenie:

indeks	kod_przedm	ocena
123456	BD2011	5.0
123456	SK2011	4.5
654321	BD2011	3.5

Student:

indeks	PESEL	nazwisko
123456	AB123456	Abacka
654321	CD345678	Babacka
987654	DE534343	Cabacka

- Zamieszczony w relacji `Zaliczenie` atrybut `indeks` służy do zidentyfikowania osoby z relacji `Student`.
- W relacji `Student` atrybut `indeks` jest kluczem.
- W relacji `Zaliczenie` atrybut `indeks` może powtarzać się lub być pusty.
- Jeśli indeks jest użyty w relacji `Zaliczenie`, to w relacji `Student` powinna występować osoba o tym indeksie (integralność referencyjna).

Więzy — podsumowanie

Więzy — podsumowanie

Więzy kolumnowe — nakładanie ograniczeń na wartość atrybutu: dziedzina, wartość nie pusta (NOT NULL), zakres;

Więzy — podsumowanie

Więzy kolumnowe — nakładanie ograniczeń na wartość atrybutu: dziedzina, wartość nie pusta (NOT NULL), zakres;

Więzy tabeli — własność klucza, unikalność w ramach tabeli;

Więzy — podsumowanie

Więzy kolumnowe — nakładanie ograniczeń na wartość atrybutu: dziedzina, wartość nie pusta (NOT NULL), zakres;

Więzy tabeli — własność klucza, unikalność w ramach tabeli;

Więzy między tabelami — własność klucza obcego;

Więzy — podsumowanie

Więzy kolumnowe — nakładanie ograniczeń na wartość atrybutu: dziedzina, wartość nie pusta (NOT NULL), zakres;

Więzy tabeli — własność klucza, unikalność w ramach tabeli;

Więzy między tabelami — własność klucza obcego;

Inne więzy ogólne — bardziej złożone warunki (np. maksymalnie dwa podejścia do przedmiotu w sesji, dostęp do wybranych przedmiotów dla studentów określonej sekcji, limit liczby osób zapisanych na zajęcia itp.)

Plan

Języki

Języki

Język definiowania danych

Musi pozwolić opisać schematy relacji oraz więzy (warunki poprawności) danych.

Języki

Język definiowania danych

Musi pozwolić opisać schematy relacji oraz więzy (warunki poprawności) danych.

Język manipulacji danymi

Pozwala dodawać/usuwać krotki z relacji.

Języki

Język definiowania danych

Musi pozwolić opisać schematy relacji oraz więzy (warunki poprawności) danych.

Język manipulacji danymi

Pozwala dodawać/usuwać krotki z relacji.

Języki zapytań

Mamy trzy propozycje:

algebra relacji — kilka operacji pozwalających działać na relacjach jako na zbiorach;

relacyjny rachunek dziedzin — język wykorzystujący formuły logiczne do opisu wartości, które należy znaleźć;

relacyjny rachunek krotek — język wykorzystujący formuły logiczne do opisu krotek, które należy znaleźć;

Języki

Język definiowania danych

Musi pozwolić opisać schematy relacji oraz więzy (warunki poprawności) danych.

Język manipulacji danymi

Pozwala dodawać/usuwać krotki z relacji.

Języki zapytań

Mamy trzy propozycje:

algebra relacji — kilka operacji pozwalających działać na relacjach jako na zbiorach;

relacyjny rachunek dziedzin — język wykorzystujący formuły logiczne do opisu wartości, które należy znaleźć;

relacyjny rachunek krotek — język wykorzystujący formuły logiczne do opisu krotek, które należy znaleźć;

Standard: **SQL**

Algebra relacji

Algebra relacji

Argumentami są całe relacje (tabele), na których wykonujemy operacje.

Algebra relacji

Argumentami są całe relacje (tabele), na których wykonujemy operacje.

Zestaw operacji jest nieliczny: rzutowanie, selekcja, iloczyn kartezjański, suma, różnica i przemianowanie

Algebra relacji

Argumentami są całe relacje (tabele), na których wykonujemy operacje.

Zestaw operacji jest nieliczny: rzutowanie, selekcja, iloczyn kartezjański, suma, różnica i przemianowanie

Zapytanie to poprawne wyrażenie algebry relacji, a odpowiedź, to wartość tego wyrażenia obliczona na podstawie aktualnego stanu bazy danych.

Operacje podstawowe - unarne

Operacje podstawowe - unarne

Rzut — $\pi_\alpha(R)$ zwraca relację o schemacie $\alpha \subseteq attr(R)$ powstałą z obcięcia relacji R do kolumn α. Na przykład $\pi_{nazwisko}(Student)$.

Student

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
765678	Cabacka	Koszalin
234565	Abacka	Legnica

Operacje podstawowe - unarne

Rzut — $\pi_\alpha(R)$ zwraca relację o schemacie $\alpha \subseteq attr(R)$ powstałą z obcięcia relacji R do kolumn α. Na przykład $\pi_{nazwisko}(Student)$. Duplikaty są eliminowane.

Student

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
765678	Cabacka	Koszalin
234565	Abacka	Legnica

Wynik rzutu na `Nazwisko`

Nazwisko
Abacka
Babacka
Cabacka
Abacka

Operacje podstawowe - unarne

Rzut — $\pi_\alpha(R)$ zwraca relację o schemacie $\alpha \subseteq attr(R)$ powstałą z obcięcia relacji R do kolumn α. Na przykład $\pi_{nazwisko}(Student)$. Duplikaty są eliminowane.

Selekcja — $\sigma_F(R)$ zwraca krotki wybrane z relacji R spełniające warunek F. Na przykład $\sigma_{Adres='Koszalin'}(Student)$.

Student

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
765678	Cabacka	Koszalin
234565	Abacka	Legnica

Operacje podstawowe - unarne

Rzut — $\pi_\alpha(R)$ zwraca relację o schemacie $\alpha \subseteq attr(R)$ powstałą z obcięcia relacji R do kolumn α. Na przykład $\pi_{nazwisko}(Student)$. Duplikaty są eliminowane.

Selekcja — $\sigma_F(R)$ zwraca krotki wybrane z relacji R spełniające warunek F. Na przykład $\sigma_{Adres='Koszalin'}(Student)$.

Warunki w selekcji - kombinacje boolowskie porównań używających operatorów $=, \neq, >, \geq, <, \leq$

Student

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
765678	Cabacka	Koszalin
234565	Abacka	Legnica

Wynik selekcji `Adres='Koszalin'`

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
765678	Cabacka	Koszalin

Operacje podstawowe - unarne

Rzut — $\pi_\alpha(R)$ zwraca relację o schemacie $\alpha \subseteq attr(R)$ powstałą z obcięcia relacji R do kolumn α. Na przykład $\pi_{nazwisko}(Student)$. Duplikaty są eliminowane.

Selekcja — $\sigma_F(R)$ zwraca krotki wybrane z relacji R spełniające warunek F. Na przykład $\sigma_{Adres='Koszalin'}(Student)$.

Przemianowanie — $\rho_{S(B_1,\ldots,B_k)}(R)$ zmienia nazwę relacji R na S i nazwy odpowiednich atrybutów R na $B_1, \ldots B_k$. Na przykład $\rho_{Osoba(id,nazwisko,miasto)}(\pi_{indeks,nazwisko,adres}(Student))$.

Student

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
765678	Cabacka	Koszalin
234565	Abacka	Legnica

Operacje podstawowe - unarne

Rzut — $\pi_\alpha(R)$ zwraca relację o schemacie $\alpha \subseteq attr(R)$ powstałą z obcięcia relacji R do kolumn α. Na przykład $\pi_{nazwisko}(Student)$. Duplikaty są eliminowane.

Selekcja — $\sigma_F(R)$ zwraca krotki wybrane z relacji R spełniające warunek F. Na przykład $\sigma_{Adres='Koszalin'}(Student)$.

Przemianowanie — $\rho_{S(B_1,\ldots,B_k)}(R)$ zmienia nazwę relacji R na S i nazwy odpowiednich atrybutów R na $B_1, \ldots B_k$. Na przykład $\rho_{Osoba(id,nazwisko,miasto)}(\pi_{indeks,nazwisko,adres}(Student))$.

Student

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
765678	Cabacka	Koszalin
234565	Abacka	Legnica

Tabela po przemianowaniu: `Osoba`

Id	Nazwisko	Miasto
123456	Abacka	Koszalin
654321	Babacka	Szczecin
765678	Cabacka	Koszalin
234565	Abacka	Legnica

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).
Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).

Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).
Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).
Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).
Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).
Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).
Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).
Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
234565	Abacka	Legnica

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).
Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).
Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres

Operacje teoriomnogościowe — suma, różnica, przekrój

Suma ($\cup$), różnica ($\setminus$), przekrój ($\cap$) — zwykłe operacje na zbiorach; $R \setminus S$ i $R \cup S$ wymagają, by $attr(R) = attr(S)$; w praktyce mogą być zastępowane operacjami na **multizbiorach** (dlaczego?).
Dodawane (odejmowane, krojone) relacje muszą mieć zgodne schematy.

StudentII

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

StudentIM

Indeks	Nazwisko	Adres
012345	Zetowski	Kielce
654321	Babacka	Szczecin

Relacja wynikowa:

Indeks	Nazwisko	Adres
654321	Babacka	Szczecin

Złączenia

Iloczyn kartezjański ($\times$) — dla relacji o rozłącznych schematach ($attr(R) \cap attr(S) = \emptyset$) $R \times S$ jest relacją o atrybutach $attr(R) \cup attr(S)$ zawierająca krotki $t = rs$, gdzie $r \in R$ i $s \in S$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Złączenia

Iloczyn kartezjański ($\times$) — dla relacji o rozłącznych schematach ($attr(R) \cap attr(S) = \emptyset$) $R \times S$ jest relacją o atrybutach $attr(R) \cup attr(S)$ zawierająca krotki $t = rs$, gdzie $r \in R$ i $s \in S$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

Przedmiot

Kod	Nazwa	Typ
BD	Bazy danych	podst
AM	Analiza mat.	obow

Złączenia

Iloczyn kartezjański ($\times$) — dla relacji o rozłącznych schematach ($attr(R) \cap attr(S) = \emptyset$) $R \times S$ jest relacją o atrybutach $attr(R) \cup attr(S)$ zawierająca krotki $t = rs$, gdzie $r \in R$ i $s \in S$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

Przedmiot

Kod	Nazwa	Typ
BD	Bazy danych	podst
AM	Analiza mat.	obow

Student $\times$ Przedmiot

Indeks	Nazwisko	Adres	Kod	Nazwa	Typ
123456	Abacka	Koszalin	BD	Bazy danych	podst
654321	Babacka	Szczecin	BD	Bazy danych	podst
234565	Abacka	Legnica	BD	Bazy danych	podst
123456	Abacka	Koszalin	AM	Analiza mat.	obow
654321	Babacka	Szczecin	AM	Analiza mat.	obow
234565	Abacka	Legnica	AM	Analiza mat.	obow

Złączenia

Iloczyn kartezjański ($\times$) — dla relacji o rozłącznych schematach ($attr(R) \cap attr(S) = \emptyset$) $R \times S$ jest relacją o atrybutach $attr(R) \cup attr(S)$ zawierająca krotki $t = rs$, gdzie $r \in R$ i $s \in S$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

Przedmiot

Kod	Nazwa	Typ
BD	Bazy danych	podst
AM	Analiza mat.	obow

Student $\times$ Przedmiot

Indeks	Nazwisko	Adres	Kod	Nazwa	Typ
123456	Abacka	Koszalin	BD	Bazy danych	podst
654321	Babacka	Szczecin	BD	Bazy danych	podst
234565	Abacka	Legnica	BD	Bazy danych	podst
123456	Abacka	Koszalin	AM	Analiza mat.	obow
654321	Babacka	Szczecin	AM	Analiza mat.	obow
234565	Abacka	Legnica	AM	Analiza mat.	obow

Złączenia

Iloczyn kartezjański ($\times$) — dla relacji o rozłącznych schematach ($attr(R) \cap attr(S) = \emptyset$) $R \times S$ jest relacją o atrybutach $attr(R) \cup attr(S)$ zawierająca krotki $t = rs$, gdzie $r \in R$ i $s \in S$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
123456	Abacka	Koszalin
654321	Babacka	Szczecin
234565	Abacka	Legnica

Przedmiot

Kod	Nazwa	Typ
BD	Bazy danych	podst
AM	Analiza mat.	obow

Student $\times$ Przedmiot

Indeks	Nazwisko	Adres	Kod	Nazwa	Typ
123456	Abacka	Koszalin	BD	Bazy danych	podst
654321	Babacka	Szczecin	BD	Bazy danych	podst
234565	Abacka	Legnica	BD	Bazy danych	podst
123456	Abacka	Koszalin	AM	Analiza mat.	obow
654321	Babacka	Szczecin	AM	Analiza mat.	obow
234565	Abacka	Legnica	AM	Analiza mat.	obow

Złączenie naturalne

Złączenie naturalne ($\bowtie$) Dla relacji R i S *złączeniem naturalnym* $R \bowtie S$ jest relacja o schemacie $attr(R) \cup attr(S)$ zawierająca krotki t, dla których istnieją krotki $r \in R$ i $s \in S$, takie że $r.(attr(R) \cap attr(S)) = s.(attr(R) \cap attr(S))$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
654321	Babacka	Szczecin
234565	Abacka	Legnica
123456	Abacka	Koszalin

Ocena

Indeks	Kod	Stopien
654321	BD	5.0
234565	BD	4.5
234565	AM	4.5
012345	AM	3.5

Złączenie naturalne

Złączenie naturalne ($\bowtie$) Dla relacji R i S *złączeniem naturalnym* $R \bowtie S$ jest relacja o schemacie $attr(R) \cup attr(S)$ zawierająca krotki t, dla których istnieją krotki $r \in R$ i $s \in S$, takie że $r.(attr(R) \cap attr(S)) = s.(attr(R) \cap attr(S))$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
654321	Babacka	Szczecin
234565	Abacka	Legnica
123456	Abacka	Koszalin

Ocena

Indeks	Kod	Stopien
654321	BD	5.0
234565	BD	4.5
234565	AM	4.5
012345	AM	3.5

Złączenie naturalne

Złączenie naturalne ($\bowtie$) Dla relacji R i S *złączeniem naturalnym* $R \bowtie S$ jest relacja o schemacie $attr(R) \cup attr(S)$ zawierająca krotki t, dla których istnieją krotki $r \in R$ i $s \in S$, takie że $r.(attr(R) \cap attr(S)) = s.(attr(R) \cap attr(S))$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
654321	Babacka	Szczecin
234565	Abacka	Legnica
123456	Abacka	Koszalin

Ocena

Indeks	Kod	Stopien
654321	BD	5.0
234565	BD	4.5
234565	AM	4.5
012345	AM	3.5

Student $\bowtie$ Ocena

Indeks	Nazwisko	Adres	Kod	Stopien

Złączenie naturalne

Złączenie naturalne ($\bowtie$) Dla relacji R i S *złączeniem naturalnym* $R \bowtie S$ jest relacja o schemacie $attr(R) \cup attr(S)$ zawierająca krotki t, dla których istnieją krotki $r \in R$ i $s \in S$, takie że $r.(attr(R) \cap attr(S)) = s.(attr(R) \cap attr(S))$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
654321	Babacka	Szczecin
234565	Abacka	Legnica
123456	Abacka	Koszalin

Ocena

Indeks	Kod	Stopien
654321	BD	5.0
234565	BD	4.5
234565	AM	4.5
012345	AM	3.5

Student $\bowtie$ Ocena

Indeks	Nazwisko	Adres	Kod	Stopien

Złączenie naturalne

Złączenie naturalne ($\bowtie$) Dla relacji R i S *złączeniem naturalnym* $R \bowtie S$ jest relacja o schemacie $attr(R) \cup attr(S)$ zawierająca krotki t, dla których istnieją krotki $r \in R$ i $s \in S$, takie że $r.(attr(R) \cap attr(S)) = s.(attr(R) \cap attr(S))$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
654321	Babacka	Szczecin
234565	Abacka	Legnica
123456	Abacka	Koszalin

Ocena

Indeks	Kod	Stopien
654321	BD	5.0
234565	BD	4.5
234565	AM	4.5
012345	AM	3.5

Student $\bowtie$ Ocena

Indeks	Nazwisko	Adres	Kod	Stopien
654321	Babacka	Szczecin	BD	5.0

Złączenie naturalne

Złączenie naturalne ($\bowtie$) Dla relacji R i S *złączeniem naturalnym* $R \bowtie S$ jest relacja o schemacie $attr(R) \cup attr(S)$ zawierająca krotki t, dla których istnieją krotki $r \in R$ i $s \in S$, takie że $r.(attr(R) \cap attr(S)) = s.(attr(R) \cap attr(S))$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
654321	Babacka	Szczecin
234565	Abacka	Legnica
123456	Abacka	Koszalin

Ocena

Indeks	Kod	Stopien
654321	BD	5.0
234565	BD	4.5
234565	AM	4.5
012345	AM	3.5

Student $\bowtie$ Ocena

Indeks	Nazwisko	Adres	Kod	Stopien
654321	Babacka	Szczecin	BD	5.0
234565	Abacka	Legnica	BD	4.5
234565	Abacka	Legnica	AM	3.5

Złączenie naturalne

Złączenie naturalne ($\bowtie$) Dla relacji R i S *złączeniem naturalnym* $R \bowtie S$ jest relacja o schemacie $attr(R) \cup attr(S)$ zawierająca krotki t, dla których istnieją krotki $r \in R$ i $s \in S$, takie że $r.(attr(R) \cap attr(S)) = s.(attr(R) \cap attr(S))$ oraz $t.attr(R) = r$ i $t.attr(S) = s$.

Student

Indeks	Nazwisko	Adres
654321	Babacka	Szczecin
234565	Abacka	Legnica
123456	Abacka	Koszalin

Ocena

Indeks	Kod	Stopien
654321	BD	5.0
234565	BD	4.5
234565	AM	4.5
012345	AM	3.5

Student $\bowtie$ Ocena

Indeks	Nazwisko	Adres	Kod	Stopien
654321	Babacka	Szczecin	BD	5.0
234565	Abacka	Legnica	BD	4.5
234565	Abacka	Legnica	AM	3.5

Krotki, które nie mają pary, nie wchodzą do wyniku!

Wszystkie operacje algebry relacji

Wszystkie operacje algebry relacji

Złączenie $\bowtie_F$ to iloczyn kartezjański połączony z selekcją:

$$R \bowtie_F S = \sigma_F(R \times S)$$

Wszystkie operacje algebry relacji

Złączenie $\bowtie_F$ to iloczyn kartezjański połączony z selekcją:

$$R \bowtie_F S = \sigma_F(R \times S)$$

Złączenia zewnętrzne to złączenie naturalne, do którego wyniku dorzuca się krotki, które nie znalazły pary. W polach, które są niewypełnione, wpisywana jest wartość NULL.

Wszystkie operacje algebry relacji

Złączenie $\bowtie_F$ to iloczyn kartezjański połączony z selekcją:

$$R \bowtie_F S = \sigma_F(R \times S)$$

Złączenia zewnętrzne to złączenie naturalne, do którego wyniku dorzuca się krotki, które nie znalazły pary. W polach, które są niewypełnione, wpisywana jest wartość NULL.

Półzłączenia to operacja wybierająca z relacji krotki, które połączyłyby się, gdyby wykonywano złączenie naturalne.

Wszystkie operacje algebry relacji

Złączenie $\bowtie_F$ to iloczyn kartezjański połączony z selekcją:

$$R \bowtie_F S = \sigma_F(R \times S)$$

Złączenia zewnętrzne to złączenie naturalne, do którego wyniku dorzuca się krotki, które nie znalazły pary. W polach, które są niewypełnione, wpisywana jest wartość NULL.

Półzłączenia to operacja wybierająca z relacji krotki, które połączyłyby się, gdyby wykonywano złączenie naturalne.

Inne operacje np. iloraz, złączenie lewostronne i prawostronne.

Wszystkie operacje algebry relacji

Złączenie $\bowtie_F$ to iloczyn kartezjański połączony z selekcją:

$$R \bowtie_F S = \sigma_F(R \times S)$$

Złączenia zewnętrzne to złączenie naturalne, do którego wyniku dorzuca się krotki, które nie znalazły pary. W polach, które są niewypełnione, wpisywana jest wartość NULL.

Półzłączenia to operacja wybierająca z relacji krotki, które połączyłyby się, gdyby wykonywano złączenie naturalne.

Inne operacje np. iloraz, złączenie lewostronne i prawostronne.

Zapytania budujemy poprawne wyrażenia używając operatorów algebry relacji, nawiasów i stałych.

Wszystkie operacje algebry relacji

Złączenie $\bowtie_F$ to iloczyn kartezjański połączony z selekcją:

$$R \bowtie_F S = \sigma_F(R \times S)$$

Złączenia zewnętrzne to złączenie naturalne, do którego wyniku dorzuca się krotki, które nie znalazły pary. W polach, które są niewypełnione, wpisywana jest wartość NULL.

Półzłączenia to operacja wybierająca z relacji krotki, które połączyłyby się, gdyby wykonywano złączenie naturalne.

Inne operacje np. iloraz, złączenie lewostronne i prawostronne.

Zapytania budujemy poprawne wyrażenia używając operatorów algebry relacji, nawiasów i stałych.

Wszystkie operacje algebry relacji są wyrażalne za pomocą: $\pi, \sigma, \rho, \times, \cup, \setminus$.

Wszystkie operacje algebry relacji

$\pi, \sigma, \rho, \times, \cup, \setminus, \bowtie$

```
\pi, \sigma, \rho, \times, \cup, \setminus, \Join
```

Kalkulator: https://dbis-uibk.github.io/relax/landing

Select DB (IMDB-sample)

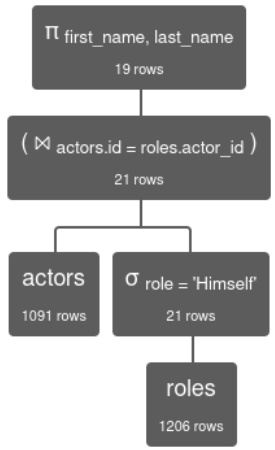
- actors
  - id number
  - first_name string
  - last_name string
  - gender string
- directors
  - id number
  - first_name string
  - last_name string
- directors_genres
  - director_id number
  - genre string
  - prob number
- movies
  - id number
  - name string
  - year number
  - rank number
- movies_directors
  - director_id number
  - movie_id number
- movies_genres
  - movie_id number
  - genre string
- roles
  - actor_id number
  - movie_id number
  - role string

Relational Algebra | SQL | Group Editor

π σ ρ ← → τ γ ∧ ∨ ¬ = ≠ ≥ ≤ ∩ ∪ ÷ - × ⋈ ⋉ ⋊ ⋋ ⋌ ▷ = -- /* {}

```
1 π first_name, last_name (actors ⋈ actors.id=roles.actor_id (σ role='Himself' (roles)))
```

execute query



π first_name, last_name (actors ⋈ actors.id = roles.actor_id (σ role = 'Himself' (roles)))

Execution time: 15 ms

actors.first_name	actors.last_name
'Mike (I)'	'Cameron'

Baza do przykładów

- **S**tudent=(indeks,nazwisko, rok), czyli indeks, nazwisko i rok studiów studenta;
- **P**rzedmiot=(nazwa, typ), czyli nazwa i typ przedmiotu;
- **O**cena=(indeks,przed,data,stop), czyli ocena uzyskana przez studenta za przedmiot wraz z datą wystawienia.

Klucze główne relacji są podkreślone. Dodatkowo w relacji *O* występują klucze obce:

- *O.indeks* odnoszący się do *S.indeks*,
- *O.przed* odnoszący się do *P.nazwa*,
- Czy pola *data* i *stop* w relacji *Ocena* mogą być puste?

Przykłady 1-3

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Przykłady 1-3

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

1. $\pi_{S.indeks, nazwisko}(\sigma_{stop=5.0 \wedge przed='BD'}(S \bowtie O))$;

Znaczenie zapytań

Przykłady 1-3

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

1. $\pi_{S.indeks, nazwisko}(\sigma_{stop=5.0 \wedge przed='BD'}(S \bowtie O))$;

Znaczenie zapytań

1. Indeksy i nazwiska studentów, którzy dostali *5.0* z *BD*.

Przykłady 1-3

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

1. $\pi_{S.indeks, nazwisko}(\sigma_{stop=5.0 \wedge przed='BD'}(S \bowtie O))$;

Znaczenie zapytań

1. Indeksy i nazwiska studentów, którzy dostali *5.0* z *BD*.
2. Pełne dane studentów, którzy dostali jakąś ocenę *5.0*.

Przykłady 1-3

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

1. $\pi_{S.indeks, nazwisko}(\sigma_{stop=5.0 \wedge przed='BD'}(S \bowtie O))$;
2. $\pi_{S.indeks, nazwisko, rok}(S \bowtie \sigma_{stop=5.0}(O))$;

Znaczenie zapytań

1. Indeksy i nazwiska studentów, którzy dostali *5.0* z *BD*.
2. Pełne dane studentów, którzy dostali jakąś ocenę *5.0*.

Przykłady 1-3

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

1. $\pi_{S.indeks, nazwisko}(\sigma_{stop=5.0 \wedge przed='BD'}(S \bowtie O))$;
2. $\pi_{S.indeks, nazwisko, rok}(S \bowtie \sigma_{stop=5.0}(O))$;

Znaczenie zapytań

1. Indeksy i nazwiska studentów, którzy dostali *5.0* z *BD*.
2. Pełne dane studentów, którzy dostali jakąś ocenę *5.0*.
3. Studenci, którzy podchodzili do *BD* co najmniej dwa razy.

Przykłady 1-3

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

1. $\pi_{S.indeks, nazwisko}(\sigma_{stop=5.0 \wedge przed='BD'}(S \bowtie O))$;
2. $\pi_{S.indeks, nazwisko, rok}(S \bowtie \sigma_{stop=5.0}(O))$;
3. $\pi_{S.indeks, nazwisko}(S \bowtie \sigma_{i1=indeks \wedge p1=przed \wedge przed='BD' \wedge data \neq d1}(\rho_{O1(i1,p1,d1,s1)}(O) \times O))$.

Znaczenie zapytań

1. Indeksy i nazwiska studentów, którzy dostali *5.0* z *BD*.
2. Pełne dane studentów, którzy dostali jakąś ocenę *5.0*.
3. Studenci, którzy podchodzili do *BD* co najmniej dwa razy.

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

4a. $\pi_{S.indeks, nazwisko, rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

Znaczenie zapytań

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

4a. $\pi_{S.indeks, nazwisko, rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

Znaczenie zapytań

4a. Studenci, którzy dostali jakąś ocenę inną niż *5.0*.

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

4a. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

4b. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

Znaczenie zapytań

4a. Studenci, którzy dostali jakąś ocenę inną niż *5.0*.

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

4a. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

4b. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

Znaczenie zapytań

4a. Studenci, którzy dostali jakąś ocenę inną niż *5.0*.

4b. Studenci, którzy nie dostali wpisu (niezgodne z więzami relacji)

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

4a. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

4b. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

4c. $\pi_{S.indeks,nazwisko,rok}(S) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop=5.0}(O))$;

Znaczenie zapytań

4a. Studenci, którzy dostali jakąś ocenę inną niż *5.0*.

4b. Studenci, którzy nie dostali wpisu (niezgodne z więzami relacji)

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

4a. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

4b. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

4c. $\pi_{S.indeks,nazwisko,rok}(S) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop=5.0}(O))$;

Znaczenie zapytań

4a. Studenci, którzy dostali jakąś ocenę inną niż *5.0*.

4b. Studenci, którzy nie dostali wpisu (niezgodne z więzami relacji)

4c. Studenci, którzy nie dostali żadnej piątki.

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

4a. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

4b. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

4c. $\pi_{S.indeks,nazwisko,rok}(S) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop=5.0}(O))$;

4d. $\pi_{S.indeks,nazwisko,rok}(S) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

Znaczenie zapytań

4a. Studenci, którzy dostali jakąś ocenę inną niż *5.0*.

4b. Studenci, którzy nie dostali wpisu (niezgodne z więzami relacji)

4c. Studenci, którzy nie dostali żadnej piątki.

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

4a. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

4b. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

4c. $\pi_{S.indeks,nazwisko,rok}(S) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop=5.0}(O))$;

4d. $\pi_{S.indeks,nazwisko,rok}(S) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

Znaczenie zapytań

4a. Studenci, którzy dostali jakąś ocenę inną niż *5.0*.

4b. Studenci, którzy nie dostali wpisu (niezgodne z więzami relacji)

4c. Studenci, którzy nie dostali żadnej piątki.

4d. Studenci, którzy mają tylko oceny *5.0* (być może nie mają żadnych).

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

4a. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

4b. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

4c. $\pi_{S.indeks,nazwisko,rok}(S) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop=5.0}(O))$;

4d. $\pi_{S.indeks,nazwisko,rok}(S) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

4e. $\pi_{S.ind,naz,rok}(S \bowtie O) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

Znaczenie zapytań

4a. Studenci, którzy dostali jakąś ocenę inną niż *5.0*.

4b. Studenci, którzy nie dostali wpisu (niezgodne z więzami relacji)

4c. Studenci, którzy nie dostali żadnej piątki.

4d. Studenci, którzy mają tylko oceny *5.0* (być może nie mają żadnych).

Przykład 4

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Studenci, którzy nie dostali 5.0.

4a. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

4b. $\pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

4c. $\pi_{S.indeks,nazwisko,rok}(S) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop=5.0}(O))$;

4d. $\pi_{S.indeks,nazwisko,rok}(S) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

4e. $\pi_{S.ind,naz,rok}(S \bowtie O) \setminus \pi_{S.indeks,nazwisko,rok}(S \bowtie \sigma_{stop \neq 5.0}(O))$;

Znaczenie zapytań

4a. Studenci, którzy dostali jakąś ocenę inną niż *5.0*.

4b. Studenci, którzy nie dostali wpisu (niezgodne z więzami relacji)

4c. Studenci, którzy nie dostali żadnej piątki.

4d. Studenci, którzy mają tylko oceny *5.0* (być może nie mają żadnych).

4e. Studenci, którzy dostają tylko piątki, przy czym bierzemy pod uwagę tylko tych, którzy mają jakikolwiek wpis.

Przykład 5

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Jak szukać czegoś, czego nie ma?

5a. $\pi_{S.indeks,nazwisko}(S) \setminus \pi_{S.indeks,nazwisko}(S \bowtie O)$;

Przykład 5

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Jak szukać czegoś, czego nie ma?

5a. $\pi_{S.indeks,nazwisko}(S) \setminus \pi_{S.indeks,nazwisko}(S \bowtie O)$;

5b. $\pi_{S.indeks,nazwisko}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

Przykład 5

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Jak szukać czegoś, czego nie ma?

5a. $\pi_{S.indeks,nazwisko}(S) \setminus \pi_{S.indeks,nazwisko}(S \bowtie O)$;

5b. $\pi_{S.indeks,nazwisko}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

5c. $\pi_{S.indeks,nazwisko}(S \bowtie \sigma_{stop=NULL}(O))$;

Przykład 5

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Jak szukać czegoś, czego nie ma?

5a. $\pi_{S.indeks,nazwisko}(S) \setminus \pi_{S.indeks,nazwisko}(S \bowtie O)$;

5b. $\pi_{S.indeks,nazwisko}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

5c. $\pi_{S.indeks,nazwisko}(S \bowtie \sigma_{stop=NULL}(O))$;

5d. $\pi_{S.indeks,nazwisko}(S \bowtie \sigma_{stop\neq NULL}(O))$;

Przykład 5

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Jak szukać czegoś, czego nie ma?

5a. $\pi_{S.indeks,nazwisko}(S) \setminus \pi_{S.indeks,nazwisko}(S \bowtie O)$;

5b. $\pi_{S.indeks,nazwisko}(S \bowtie \sigma_{stop\ IS\ NULL}(O))$;

5c. $\pi_{S.indeks,nazwisko}(S \bowtie \sigma_{stop=NULL}(O))$;

5d. $\pi_{S.indeks,nazwisko}(S \bowtie \sigma_{stop\neq NULL}(O))$;

Krotka jest wybierana przez selekcję, gdy warunek ma dla niej wartość TRUE. Wartość UNKNOWN nie wystarcza.

Przykład 6

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Można pytać o to samo na różne sposoby. Czy to ma jakieś znaczenie?

(6)
$$\pi_{nazwisko,indeks}($$
$$\quad \sigma_{stop=5.0 \wedge typ='' zaaw''}(\sigma_{nazwa=przed}(P \times O)) \bowtie$$
$$\quad \sigma_{rok=4}(S))$$
$$\cup \pi_{nazwisko,indeks}($$
$$\quad \sigma_{stop=5.0 \wedge typ='' obow''}(\sigma_{nazwa=przed}(P \times O)) \bowtie$$
$$\quad\quad \sigma_{rok=3}(S));$$

(6a)
$$\pi_{nazwisko,indeks}($$
$$\sigma_{((rok=3 \wedge typ='obow') \vee (rok=4 \wedge typ='zaaw'))}(\sigma_{rok=3 \vee rok=4}(S) \bowtie$$
$$\pi_{indeks,typ}(\rho_{P(przed,typ)}(\sigma_{typ='zaaw' \vee typ='obow'}(P))) \bowtie$$
$$\pi_{indeks,przed}(\sigma_{stop=5.0}(O)))))$$

Przykład 7 - poszukajmy najlepszych z BD

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

Przykład 7 - poszukajmy najlepszych z BD

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

(7a) $\pi_{indeks}(\sigma_{stop>s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

Znaczenie zapytań

Przykład 7 - poszukajmy najlepszych z BD

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

(7a) $\pi_{indeks}(\sigma_{stop>s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

Znaczenie zapytań

7a. Indeksy studentów, którzy z BD mają ocenę lepszą niż ktoś inny, czyli nie są najgorsi.

Przykład 7 - poszukajmy najlepszych z BD

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

(7a) $\pi_{indeks}(\sigma_{stop>s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

Znaczenie zapytań

7a. Indeksy studentów, którzy z BD mają ocenę lepszą niż ktoś inny, czyli nie są najgorsi.

7b. Indeksy studentów, którzy z BD mają ocenę gorszą niż ktoś inny, czyli nie są najlepsi (dopełnienie tego, czego szukamy).

Przykład 7 - poszukajmy najlepszych z BD

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

(7a) $\pi_{indeks}(\sigma_{stop>s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7b) $\pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

Znaczenie zapytań

7a. Indeksy studentów, którzy z BD mają ocenę lepszą niż ktoś inny, czyli nie są najgorsi.

7b. Indeksy studentów, którzy z BD mają ocenę gorszą niż ktoś inny, czyli nie są najlepsi (dopełnienie tego, czego szukamy).

Przykład 7 - poszukajmy najlepszych z BD

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

(7a) $\pi_{indeks}(\sigma_{stop>s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7b) $\pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7c) $\pi_{indeks}(S) \setminus \pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

Znaczenie zapytań

7a. Indeksy studentów, którzy z BD mają ocenę lepszą niż ktoś inny, czyli nie są najgorsi.

7b. Indeksy studentów, którzy z BD mają ocenę gorszą niż ktoś inny, czyli nie są najlepsi (dopełnienie tego, czego szukamy).

Przykład 7 - poszukajmy najlepszych z BD

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

(7a) $\pi_{indeks}(\sigma_{stop>s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7b) $\pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7c) $\pi_{indeks}(S) \setminus \pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

Znaczenie zapytań

7a. Indeksy studentów, którzy z BD mają ocenę lepszą niż ktoś inny, czyli nie są najgorsi.

7b. Indeksy studentów, którzy z BD mają ocenę gorszą niż ktoś inny, czyli nie są najlepsi (dopełnienie tego, czego szukamy).

7c. Indeksy studentów, którzy nie są od nikogo gorsi z BD.

Przykład 7 - poszukajmy najlepszych z BD

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

(7a) $\pi_{indeks}(\sigma_{stop>s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7b) $\pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7c) $\pi_{indeks}(S) \setminus \pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7d) $\pi_{indeks}(\sigma_{przed=''BD''}(O)) \setminus \pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

Znaczenie zapytań

7a. Indeksy studentów, którzy z BD mają ocenę lepszą niż ktoś inny, czyli nie są najgorsi.

7b. Indeksy studentów, którzy z BD mają ocenę gorszą niż ktoś inny, czyli nie są najlepsi (dopełnienie tego, czego szukamy).

7c. Indeksy studentów, którzy nie są od nikogo gorsi z BD.

Przykład 7 - poszukajmy najlepszych z BD

Baza danych

$$S = (\underline{indeks}, nazwisko, rok),\ P = (\underline{nazwa}, typ),\ O = (\underline{indeks}, \underline{przed}, \underline{data}, stop)$$

(7a) $\pi_{indeks}(\sigma_{stop>s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7b) $\pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7c) $\pi_{indeks}(S) \setminus \pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

(7d) $\pi_{indeks}(\sigma_{przed=''BD''}(O)) \setminus \pi_{indeks}(\sigma_{stop<s1 \wedge przed=''BD'' \wedge p1=przed}(O \bowtie \rho_{O1(i1,p1,d1,s1)}(O)))$

Znaczenie zapytań

7a. Indeksy studentów, którzy z BD mają ocenę lepszą niż ktoś inny, czyli nie są najgorsi.

7b. Indeksy studentów, którzy z BD mają ocenę gorszą niż ktoś inny, czyli nie są najlepsi (dopełnienie tego, czego szukamy).

7c. Indeksy studentów, którzy nie są od nikogo gorsi z BD.

7d. Indeksy studentów, którzy są najlepsi z BD.

Wnioski i uwagi:

1. Algebra relacji jest językiem imperatywnym (operacyjnym).

Wnioski i uwagi:

1. Algebra relacji jest językiem imperatywnym (operacyjnym).
2. Znaczenie zapytania (w języku naturalnym) nie zawsze jest oczywiste, gdyż algebra relacji nie przypomina języka naturalnego.

Wnioski i uwagi:

1. Algebra relacji jest językiem imperatywnym (operacyjnym).
2. Znaczenie zapytania (w języku naturalnym) nie zawsze jest oczywiste, gdyż algebra relacji nie przypomina języka naturalnego.
3. To samo zapytanie może mieć wiele równoważnych postaci — mogą one różnić się złożonością wykonania.

Wnioski i uwagi:

1. Algebra relacji jest językiem imperatywnym (operacyjnym).
2. Znaczenie zapytania (w języku naturalnym) nie zawsze jest oczywiste, gdyż algebra relacji nie przypomina języka naturalnego.
3. To samo zapytanie może mieć wiele równoważnych postaci — mogą one różnić się złożonością wykonania.
4. Na podstawie samego opisu trudno określić moc tego języka.

Wnioski i uwagi:

1. Algebra relacji jest językiem imperatywnym (operacyjnym).
2. Znaczenie zapytania (w języku naturalnym) nie zawsze jest oczywiste, gdyż algebra relacji nie przypomina języka naturalnego.
3. To samo zapytanie może mieć wiele równoważnych postaci — mogą one różnić się złożonością wykonania.
4. Na podstawie samego opisu trudno określić moc tego języka.
5. Algebra relacji jest podstawą SQL.

Materiały na skosie:

1. Więcej przykładów zapytań w algebrze relacji:
   `https://skos.ii.uni.wroc.pl/pluginfile.php/68435/mod_label/intro/algebra_relacji_trudne_przyklady_EU.pdf`

Materiały na skosie:

1. Więcej przykładów zapytań w algebrze relacji:
   `https://skos.ii.uni.wroc.pl/pluginfile.php/68435/mod_label/intro/algebra_relacji_trudne_przyklady_EU.pdf`
2. Kalkulator algebry relacji:
   `https://dbis-uibk.github.io/relax/landing`