

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



Predmet: Osnove podatkovnih baz

Modul:
Poizvedovalni jezik SQL

Gradivo:
v.2017



Vsebina

- Splošno o SQL ter zgodovina razvoja
- Sintaksa SQL
 - Ukazi iz skupine DML: `SELECT`, `INSERT`, `UPDATE`, `DELETE`
 - Ukazi iz skupine DDL: `CREATE`, `ALTER`, `SET TRANSACTION`, ...
 - Sprožilci, funkcije in shranjeni programi
- Dostopanje do MySQL iz programskih jezikov



Uvod v SQL...

- SQL sestavljata dve skupini ukazov:
 - Skupina ukazov za **DDL** (*Data Definition Language*) za opredelitev strukture podatkovne baze in
 - Skupina ukazov za **DML** (*Data Manipulation Language*) za poizvedovanje in ažuriranje podatkov.

```
create table oseba (  
    EMŠO number (13),  
    ime char(20),  
    priimek char(20)  
)
```

```
select ime, priimek  
from oseba  
where ime = 'Tine'  
order by priimek
```

- Do 1999 SQL brez ukazov za **kontrolni tok** - potrebno obvladati s programskim jezikom ali interaktivno.



Uvod v SQL...

- Lastnosti SQL:
 - Enostaven;
 - Nepostopkoven (kaj in ne kako);
 - Uporaben v okviru številnih vlog: skrbniki PB, vodstvo, razvijalci informacijskih rešitev, končni uporabniki...;
 - SQL de-facto in tudi uradno standardni jezik za delo z relacijskimi podatkovnimi bazami.



Zgodovina SQL

- V 1970h IBM razvija sistem *System R*, ki bo temeljil na relacijskem modelu.
- 1974 – D. Chamberlin in F. Boyce (*IBM San Jose Laboratory*) definirata jezik ‘**Structured English Query Language**’ (SEQUEL).
 - SEQUEL se kasneje preimenuje v SQL
- V poznih 1970h – *Relational Software* (danes *Oracle*) razvije svoj SUPB, ki temelji na relacijskem modelu in implementira SQL.
- Poleti 1979 – Oracle izda prvo komercialno različico SQL; nekaj tednov pred IBM-ovo implementacijo *System/38*



Standardizacija SQL

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First formalized by ANSI.
1989	SQL-89	FIPS 127-1	Minor revision, in which the major addition were integrity constraints. Adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), <i>Entry Level</i> SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries (e.g. transitive closure), triggers , support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features (e.g. structured types). Support for embedding SQL in Java (SQL/OLB) and vice-versa (SQL/JRT).
2003	SQL:2003	SQL 2003	Introduced XML -related features (SQL/XML), <i>window functions</i> , standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006	SQL 2006	ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form. In addition, it enables applications to integrate into their SQL code the use of XQuery , the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents. ^[33]
2008	SQL:2008	SQL 2008	Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers. Adds the TRUNCATE statement. ^[34]
2011	SQL:2011		

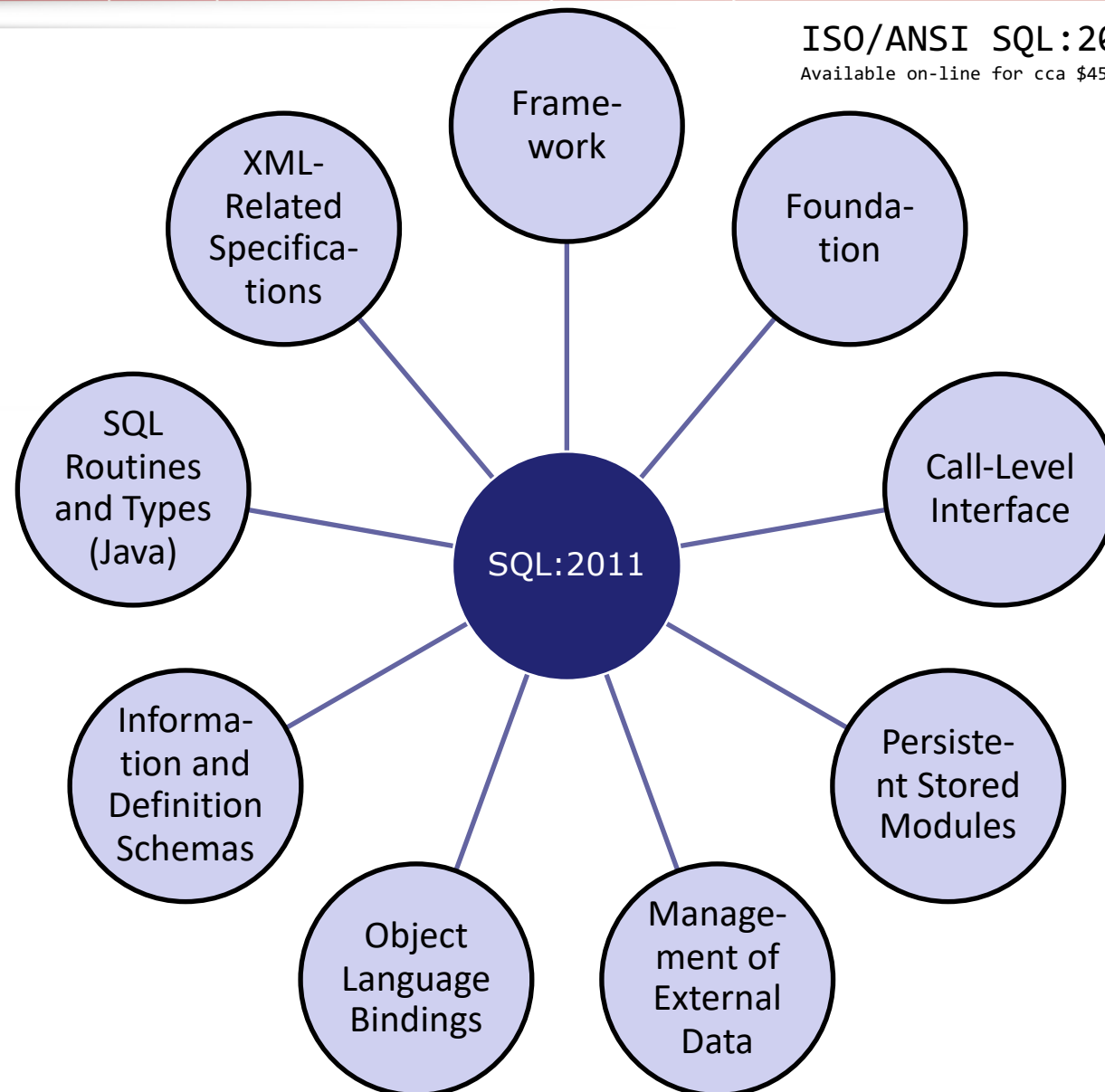
Vir: Wikipedia EN



SQL:2011

ISO/ANSI SQL:2011 standard

Available on-line for cca \$450





Implementacije SQL

- Med standardi SQL-92, SQL:1999; SQL:2008,... razlike
- Implementacije ponudnikov SUPB različne (dialekti)
- Primerjava implementacij SQL
<http://troels.arvin.dk/db/rdbms/>



Stavki skupine SQL DML

- **DML skupina** zajema SQL stavke za manipulacijo s podatki
 - SELECT → Izbira
 - INSERT → Dodajanje
 - DELETE → Brisanje
 - UPDATE → Spreminjanje
- Sintaksa SELECT stavka najbolj kompleksna



SELECT stavek...

SELECT [DISTINCT | ALL]

{* | [columnExpression [AS newName]] [,...]}

FROM TableName [alias] [, ...]

[WHERE condition]

[GROUP BY columnList]

[HAVING condition]

[ORDER BY columnList]



SELECT stavek...

- **SELECT** Določa stolpce, ki naj se pojavijo v izhodni relaciji
- **FROM** Določa tabele za poizvedbo
- **WHERE** Filtrira vrstice
- **GROUP BY** Združuje vrstice po vrednostih izbranih stolpcev
- **HAVING** Filtrira skupine glede na določene pogoje
- **ORDER BY** Določa vrstni red vrstic na izhodu

**Vrstnega reda sklopov ni možno spreminjati!
Obvezna sta samo SELECT in FROM sklopa!**



Primeri

- SUPB: **MySQL**
- Testni podatki:
 - Employees.db
 - <https://launchpad.net/test-db/+download>
 - Tabele:
 - Employees
 - Salaries
 - Titles
 - Dept_emp
 - Departments
 - Dept_manager
 - Skupaj 3.919.015 zapisov



Primeri

Ali je shema popolna? Ji kaj manjka?

- Za primere bomo uporabljali shemo **employees**

`employees(emp_no, birth_date, first_name, last_name, gender,
hire_date)`

`titles(emp_no, title, from_date, to_date)`

`salaries(emp_no, salary, from_date, to_date)`

`dept_emp(emp_no, dept_no, from_date, to_date)`

`departments(dept_no, dept_name)`

`dept_manager(dept_no, emp_no, from_date, to_date)`



Primeri

- Za primere bomo uporabljali shemo **employees**

`employees(emp_no, birth_date, first_name, last_name, gender,
hire_date)`

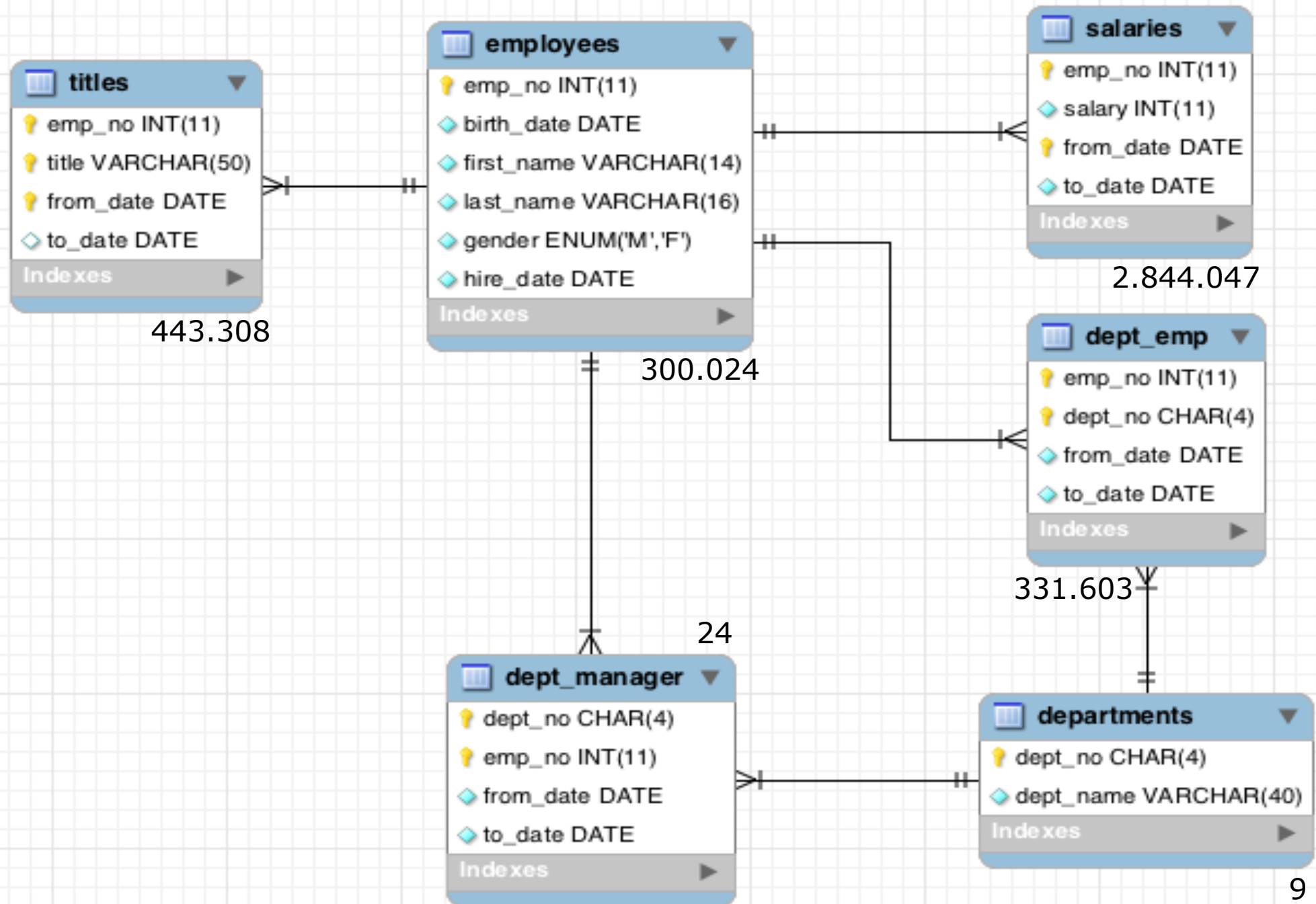
`titles(#emp_no, title, from_date, to_date)`

`salaries(#emp_no, salary, from_date, to_date)`

`dept_emp(#emp_no, #dept_no, from_date, to_date)`

`departments(dept_no, dept_name)`

`dept_manager(#dept_no, #emp_no, from_date, to_date)`



employees

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1986-06-26	Piveteau	Piveteau	F	1989-08-24
			Sluis	F	1990-01-22
			Brideland	M	1992-12-18

Salaries

emp_no	salary	from_date	to_date
10001	60117	1986-06-26	1987-06-26
10001	62102	1987-06-26	1988-06-25
10001	66074	1988-06-25	1989-06-25
10001	66596	1989-06-25	1990-06-25
10001	66961	1990-06-25	1991-06-25
10001	71046	1991-06-25	1992-06-24
10001	74333	1992-06-24	1993-06-24

titles

emp_no	title	from_date	to_date
10001	Senior Engineer	1986-06-26	9999-01-01
10002	Staff	1996-08-03	9999-01-01
10003	Senior Engineer	1995-12-03	9999-01-01
10004	Engineer	1986-12-01	1995-12-01
10004	Senior Engineer	1995-12-01	9999-01-01
10005	Senior Staff	1996-09-12	9999-01-01
10005	Staff	1989-09-12	1996-09-12
10006	Senior Engineer	1990-08-05	9999-01-01

departments

dept_no	dept_name
d009	Customer Service
d005	Development
d002	Finance
d003	Human Resources
d001	Marketing
d004	Production
d006	Quality Management
d008	Research
d007	Sales

MySQL Workbench

Admin (mysql@localhost) SQL Editor (LocalServer)

SCHEMAS

Search objects

- Citations
- employees
 - Tables
 - departments
 - dept_emp
 - dept_manager
 - employees
 - Columns
 - emp_no
 - birth_date
 - first_name
 - last_name
 - gender
 - hire_date
 - Indexes
 - Foreign Keys
 - Triggers
 - salaries
 - titles
 - Views
 - Routines
- HotelChain

Object Info Session

No object selected

SQL File 1 x SQL File 2* x

```

1 use employees;
2 select * from employees;

```

100% 24:2

Filter: Search Edit: Export: Fetch rows:

emp_no	birth_date	first_name	last_name	gender	hire_date
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
10011	1953-11-07	Mary	Sluis	F	1990-01-22
10012	1960-10-04	Patricio	Bridgland	M	1992-12-18
10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20
10014	1956-02-12	Berni	Genin	M	1987-03-11

employees 2 Apply Revert

Action Output

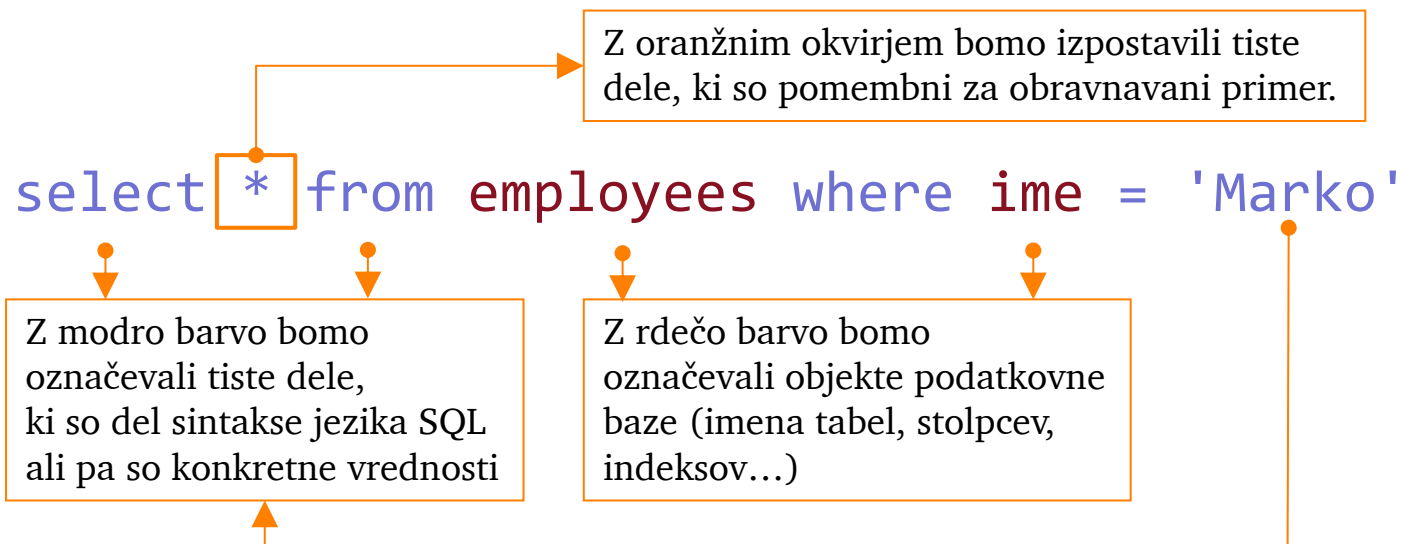
	Time	Action	Response	Duration / Fetch Time
✓ 1	00:57:26	use employees	0 row(s) affected	0.001 sec
✓ 2	00:57:39	select * from employees limit 5	5 row(s) returned	0.001 sec / 0.000 sec
✓ 3	10:57:48	select * from employees LIMIT 0, 1000	1000 row(s) returned	0.001 sec / 0.001 sec

Query Completed



Pravila formatiranja SQL primerov

- SQL izrazi so neobčutljivi na velikost črk.
- Barve niso predpisane.
- Rezervirane besede SQL ne smemo uporabljati za definicije podatkovnih objektov (tabel, stolpcev itn.)
- Sintaksa, ki jo bomo uporabljali za primere:





Enostaven izpis...

- Izpiši vse podatke o zaposlenih

```
select * from employees;
```

emp_no	birth_date	first_name	last_name	gender	hire_date
▶ 10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezael	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
10011	1953-11-07	Mary	Sluis	F	1990-01-22
10012	1960-10-04	Patricio	Bridgland	M	1992-12-18
10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20
10014	1956-02-12	Berni	Genin	M	1987-03-11
10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02



Enostaven izpis...

- Izpiši imena in priimke zaposlenih

```
select first_name, last_name from employees;
```

first_name	last_name
Georgi	Facello
Bezalel	Simmell
Parto	Bamford
Chirstian	Koblick
Kyoichi	Maliniak
Anneke	Preusig
Tzvetan	Zielinski
Saniya	Kalloufi
Sumant	Peac
Duangkaew	Piveteau
Mary	Sluis
Patricio	Bridgland
Eberhardt	Terkki
Berni	Genin
Guoxiang	Nooiteboom



Uporaba DISTINCT

- Izpiši vse različne vrednosti, ki nastopajo v stolpcu

```
select distinct gender  
from employees;
```

gender
M
F



Izračunana polja in funkcije...

- Izpiši starost delavcev ob zaposlitvi

```
select first_name, last_name, birth_date, hire_date,  
DATEDIFF(hire_date, birth_date)/365  
as 'Starost ob zaposlitvi'  
from employees;
```

first_name	last_name	birth_date	hire_date	Starost ob zaposlitvi
Georgi	Facello	1953-09-02	1986-06-26	32.8356
Bezalel	Simmell	1964-06-02	1985-11-21	21.4849
Parto	Bamford	1959-12-03	1986-08-28	26.7534
Chirstian	Koblick	1954-05-01	1986-12-01	32.6082
Kyoichi	Molinski	1955-01-21	1980-09-12	24.6658



Izračunana polja in funkcije...

- Druga možnost...

```
select first_name, last_name,  
YEAR(hire_date)-YEAR(birth_date)  
as 'Starost ob zaposlitvi'  
from employees;
```

first_name	last_name	Starost ob zaposlitvi
Georgi	Facello	33
Bezalel	Simmel	21
Parto	Bamford	27
Chirstian	Koblick	32
Kyoichi	Maliniak	34



Funkcije

■ Primer: [MySQL](#), [Date&Time](#) functions

Name	Description
ADDDATE ()	Add time values (intervals) to a date value
ADDTIME ()	Add time
CONVERT_TZ ()	Convert from one timezone to another
CURDATE ()	Return the current date
CURRENT_DATE () , CURRENT_DATE	Synonyms for CURDATE()
CURRENT_TIME () , CURRENT_TIME	Synonyms for CURTIME()
CURRENT_TIMESTAMP () , CURRENT_TIMESTAMP	Synonyms for NOW()
CURTIME ()	Return the current time
DATE_ADD ()	Add time values (intervals) to a date value
DATE_FORMAT ()	Format date as specified
DATE_SUB ()	Subtract a time value (interval) from a date
DATE ()	Extract the date part of a date or datetime expression
DATEDIFF ()	Subtract two dates
DAY ()	Synonym for DAYOFMONTH()
DAYNAME ()	Return the name of the weekday



Iskalni kriteriji

- Izpiši vse zaposlene, ki so moškega spola in so rojeni pred 1953.

```
select * from employees
```



```
YEAR(birth_date)<1953 and  
gender = 'M';
```

emp_no	birth_date	first_name	last_name	gender	hire_date
▶ 10020	1952-12-24	Mayuko	Warwick	M	1991-01-26
10022	1952-07-08	Shahaf	Famili	M	1995-08-22
10047	1952-06-29	Zvonko	Nyanchama	M	1989-03-31
10066	1952-11-13	Kwee	Schusler	M	1986-02-26
10097	1952-02-27	Remzi	Waschkowski	M	1990-09-15
10106	1952-08-29	Eben	Aingworth	M	1990-12-19
10108	1952-04-07	Lunin	Civon	M	1986-10-02



Iskanje z uporabo BETWEEN

- Izpiši zaposlene rojene med 30. in 31. decembrom 1955.

```
select * from employees where birth_date  
between '1955-12-30' and '1955-12-31';
```

emp_no	birth_date	first_name	last_name	gender	hire_date
12715	1955-12-31	Honglan	Showalter	M	1991-07-31
19386	1955-12-30	Leandro	Peltason	F	1987-04-01
20437	1955-12-30	Sanjit	Versino	F	1989-09-06
21401	1955-12-31	Uzi	Vakili	F	1992-09-21
25393	1955-12-30	Claudi	Frijda	F	1987-07-12
25562	1955-12-30	Bernardo	Menhardt	F	1986-11-16
26599	1955-12-30	Katsun	Armand	M	1987-07-01



Iskanje po članstvu množice

- Izpiši podatke o delovnih mestih, kjer gre za delovno mesto **staff** ali **senior staff**.

```
select * from titles  
where title in ('Staff', 'Senior Staff');
```

emp_no	title	from_date	to_date
10002	Staff	1996-08-03	9999-01-01
10005	Senior Staff	1996-09-12	9999-01-01
10005	Staff	1989-09-12	1996-09-12
10007	Senior Staff	1996-02-11	9999-01-01
10007	Staff	1989-02-10	1996-02-11
10011	Staff	1990-01-22	1996-11-09
10013	Senior Staff	1985-10-20	9999-01-01
10015	Senior Staff	1997-08-10	1997-08-22



Iskanje z vzorcem

- Izpiši vse zaposlene, ki se pišejo na 'B' in so se rodili prvega v mesecu v šestdesetih letih.

```
select * from employees  
where last_name like 'B%' and  
       birth_date like '__6___01';
```

emp_no	birth_date	first_name	last_name	gender	hire_date
10056	1961-09-01	Brendon	Bernini	F	1990-02-01
13014	1962-06-01	Gianluca	Budinsky	M	1986-01-21
13199	1962-03-01	Gunilla	Bergere	M	1986-04-17
14499	1963-09-01	Vasilis	Bernardinello	F	1992-02-12
15882	1960-05-01	Gao	Bazelow	F	1990-11-03
15890	1962-09-01	Feiyu	Bratsberg	M	1991-07-26
16936	1964-10-01	Vitaly	Baar	F	1989-12-08
17180	1963-05-01	Salvador	Boguraev	M	1987-05-15
19862	1962-04-01	Marla	Bouloucos	M	1991-11-04
20379	1962-06-01	Arve	Bain	M	1989-11-08



Iskanje z NULL vrednostjo v pogoju

- Izpiši vse zaposlene, ki nimajo podatka o datumu zaposlitve!

```
select * from employees  
where hire_date is null;
```

emp_no	birth_date	first_name	last_name	gender	hire_date
▶	NULL	NULL	NULL	NULL	NULL



Sortiranje vrstic v izhodni relaciji

- Izpiši podatke o zaposlenih sortirane po priimku naraščujoče in po datumu rojstva padajoče.

```
select emp_no, last_name, birth_date  
from employees  
order by last_name, birth_date desc;
```

emp_no	last_name	birth_date
234739	Aamodt	1965-01-31
100916	Aamodt	1965-01-18
22105	Aamodt	1964-12-10
450068	Aamodt	1964-12-07
448061	Aamodt	1964-12-01
227725	Aamodt	1964-11-26
463875	Aamodt	1964-10-24
11761	Aamodt	1964-07-17
295537	Aamodt	1964-06-30



Agregiranje podatkov...

- ISO standard definira pet agregarnih operacij
 - **COUNT** vrne število vrednosti v določenem stolpcu
 - **SUM** vrne seštevek vrednosti v določenem stolpcu
 - **AVG** vrne povprečje vrednosti v določenem stolpcu
 - **MIN** vrne najmanjšo vrednost v določenem stolpcu
 - **MAX** vrne največjo vrednost v določenem stolpcu
- Vse operacije delujejo na enem stolpcu in vračajo eno samo vrednost.



Agregiranje podatkov...

- **COUNT**, **MIN** in **MAX** se uporabljajo za numerične in ne-numerične vrednosti, **SUM** in **AVG** zahtevata numerične vrednosti.
- Vse operacije razen **COUNT(*)** najprej odstranijo vrstice z **NULL vrednostjo** v stolpcu, po katerem agregiramo.
- **COUNT(*)** prešteje vse vrstice, ne glede na **NULL** vrednosti ali duplikate.



Agregiranje podatkov...

- Če se želimo znebiti duplikatov, uporabimo **DISTINCT** pred imenom stolpca.
- **DISTINCT** nima učinka na **MIN/MAX**, lahko pa vpliva na **SUM/AVG**.
- Agregarne operacije lahko uporabimo le v **SELECT** ali **HAVING** sklopu

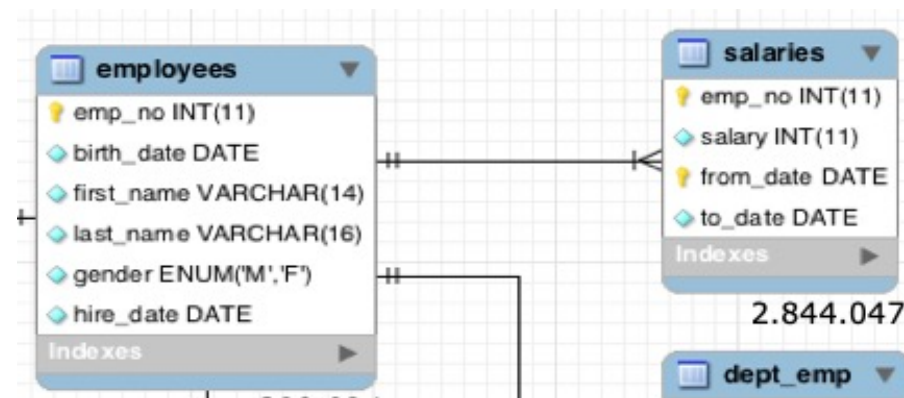


Agregiranje podatkov

- Če **SELECT** sklop vsebuje agregarno operacijo, mora obstajati tudi **GROUP BY** sklop, sicer ni moč dodeliti agregirane vrednosti.

```
select emp_no, avg(salary)
from salaries;
```

emp_no	avg(salary)
10001	63810.7448





Agregiranje podatkov

■ ...

```
select emp_no, avg(salary)
from salaries
group by emp_no;
```

emp_no	avg(salary)
10001	75388.9412
10002	68854.5000
10003	43030.2857
10004	56512.2500
10005	87275.7692
10006	50514.9167
10007	70826.7143
10008	49307.6667
10009	78284.5556

Če uporabimo še
order by avg(salary)

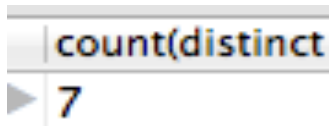
emp_no	avg(salary)
15830	39299.5000
253406	39332.7500
473390	39372.5000
281546	39417.2500
466226	39442.7500
401786	39453.7500
245832	39515.2500
230890	39520.0000
406107	39572.2222



Uporaba COUNT in DISTINCT

- Koliko različnih delovnih mest obstaja v tabeli **titles**?

```
select count(distinct title)  
from titles;
```



A screenshot of a SQL query result. The query is 'count(distinct title)' and the result is '7'. The text is displayed in a monospaced font, with the result '7' highlighted in blue.



Uporaba več agregatov istočasno

- Izpiši minimalno, maksimalno in povprečno plačo.

```
select min(salary), max(salary), avg(salary)  
from salaries;
```

min(salary)	max(salary)	avg(salary)
38623	158220	63810.7448



Združevanje podatkov...

- **GROUP BY** : združevanje podatkov v skupine.
- Ukaza **SELECT** in **GROUP BY** tesno povezana

```
select emp_no, avg(salary)
from salaries group by emp_no;
```

emp_no	salary
10001	60117
10001	62102
10001	66074
10001	66596
10001	66961
10001	71046
10001	74333
10001	75286
10001	75994
10001	76884

emp_no	avg(salary)
10001	75388.9412
10002	68854.5000
10003	43030.2857
10004	56512.2500
10005	87275.7692
10006	50514.9167
10007	70826.7143
10008	49307.6667
10009	78284.5556



Združevanje podatkov...

■ Pravila:

- Stolpci v **SELECT** tudi v **GROUP BY** (izjema stolpci, ki nastopajo samo v agregarnih operacijah).
- Najprej se izvede **WHERE**, združevanje na preostalih vrsticah.
- Pri združevanju **NULL** vrednosti tretirane kot enake.

Kaj vrne ta poizvedba?

```
Select YEAR(birth_date) as 'Rojeni leta',  
       count(*) as 'Število'  
from employees  
where gender = 'F'  
group by YEAR(birth_date);
```

Rojeni leta	Število
1952	8502
1953	9104
1954	9385
1955	9267
1956	9229
1957	9069
1958	9312
1959	9286
1960	9383
1961	9269



Omejitev skupin

- **HAVING** sklop postavlja pogoje, katerim morajo zadoščati skupine v rezultatu.
 - ! Stolpci v **HAVING** sklopu morajo biti tudi v **SELECT** sklopu ali agregatih.

```
select YEAR(birth_date), count(*)  
from employees  
where gender = 'F'  
group by YEAR(birth_date)  
having count(*) > 9200;
```

YEAR(birth_date)	count(*)
1954	9385
1955	9267
1956	9229
1958	9312
1959	9286
1960	9383
1961	9269
1962	9216



Gnezdenje poizvedb...

- Vgnezdene **SELECT** stavki se lahko uporabijo v **WHERE** ali **HAVING** sklopih drugega **SELECT** stavka (*subselect*).

```
select first_name, last_name
from employees
where emp_no in (
    select emp_no
    from titles
    where title = 'manager'
);
```

first_name	last_name
Margareta	Markovitch
Vishwani	Minakawa
Ebru	Alpin
Isamu	Legleitner
Shirish	Ossenbruggen
Karsten	Sigstam
Krassimir	Wegerle
Rosine	Cools
Shem	Kieras
Oscar	Ghazalie
DeForest	Hagimont
Leon	DasSarma
Paternela	Onuiche



Primer vgnezdenega stavka...WHERE

- Izpiši imena in priimke zaposlenih, ki so delali na vsaj treh delovnih mestih.

```
select first_name, last_name  
from employees  
where emp_no in (  
    select emp_no  
    from titles  
    group by emp_no  
);
```





Primer vgnezdenega stavka ... FROM

- Izpiši najvišjo povprečno plačo.

```
select max(dE.avgSalary)
from (
  select avg(salary) as avgSalary
  from salaries
  group by emp_no
) dE;
```

max(dE.avgSalary)
141835.3333



Pravila gnezdenja SELECT stavkov...

- Pravila vgnezenih **SELECT** stavkov...:
 - Uporaba **ORDER BY** v vgnezenem stavku nesmiselna.
 - **SELECT** vgnezenega stavka lahko zajema samo en stolpec (razen v primeru uporabe ukaza **EXISTS**.
 - Mnoge implementacije tega ne upoštevajo. Tudi MySQL.

```
select... where (emp_no, title) in (  
    select emp_no, title from...  
);
```



Pravila gnezdenja SELECT stavkov...

- Pravila vgnezenih **SELECT** stavkov...:
 - Imena stolpcev v vgnezenem stavku se nanašajo na tabele iz vgnezenega ali zunanjega stavka (uporaba alias-ov)

```
select E.emp_no, E.first_name, E.last_name
from employees E
where E.emp_no in (
    select S.emp_no
    from salaries S
    where E.birth_date >= S.from_date
);
```



Pravila gnezdenja SELECT stavkov...

- Pravila vgnezenih **SELECT** stavkov...:
 - Ko je vgnezen SELECT stavek operand v primerjavi, se mora nahajati na desni strani enačbe.
 - Ne velja povsod. Tudi v MySQL ne.

```
select first_name, last_name  
from employees  
where emp_no in (  
    select emp_no  
    from salaries  
    where (select max(salary)  
           from salaries) = salary);
```

first_name	last_name
Tokuyasu	Pesch



Pravila gnezdenja SELECT stavkov...

- Pravila vgnezenih **SELECT** stavkov...:
 - Vgnezdeni **SELECT** stavek ne more biti operand v izrazu.
 - Ne velja povsod. Tudi v MySQL ne.

```
select emp_no,  
       avg(salary) - (select avg(salary) from salaries)  
from salaries  
group by emp_no;
```

emp_no	avg(salary) - (s...
10001	11578.1963
10002	5043.7552
10003	-20780.4591
10004	-7298.4948
10005	23465.0244
10006	-13295.8282
10007	7015.9694
10008	-14503.0782
10009	14473.8107
10010	12912.2552
10011	-14078.7448



Uporaba ANY/SOME, ALL

- V vgnezenih stavkih, ki vračajo en sam stolpec, lahko uporabljamo operatorja **ANY/SOME** in **ALL**.
 - **ANY/SOME**: pogoj izpolnjen, če valja za vsaj eno vrednosti poizvedbe.
 - **ALL**: pogoj izpolnjen, če valja za vse vrednosti poizvedbe.
- Če rezultat poizvedbe prazen:
 - **ALL** vrne **true**,
 - **ANY/SOME** vrne **false**.



Primer uporabe ANY

- Izpiši podatke o zaposlenih, za katere velja, da se pred njimi ni rodil nobeden od ostalih zaposlenih.

```
select emp_no, first_name, last_name, birth_date
from employees
where birth_date <=
  ALL(select birth_date from employees);
```

emp_no	first_name	last_name	birth_date
65308	Jouni	Pocchiola	1952-02-01
87461	Moni	Decaestecker	1952-02-01
91374	Eishiro	Kuzuoka	1952-02-01
207658	Kiyokazu	Whitcomb	1952-02-01
237571	Ronghao	Schaad	1952-02-01
406121	Supot	Remmele	1952-02-01



Uporaba EXISTS

- **EXISTS** lahko uporabljamo le v vgnezdenih poizvedbah.
- Vrača logičen rezultat **true/false**.
 - **True**: če obstaja vsaj ena vrstica v tabeli, ki je rezultat vgnezdene poizvedbe.
 - **False**: če vgnezdena poizvedba vrača prazno množico.



Primer uporabe EXISTS

```
select E.emp_no, E.first_name, E.last_name, E.hire_date
from employees E where not exists (
    select E1.emp_no
    from employees E1
    where E1.hire_date < E.hire_date
);
```

	emp_no	first_name	last_name	hire_date
▶	110022	Margareta	Markovitch	1985-01-01
	110085	Ebru	Alpin	1985-01-01
	110183	Shirish	Ossenbruggen	1985-01-01
	110303	Krassimir	Wegerle	1985-01-01
	110511	DeForest	Hagimont	1985-01-01
	110725	Peternela	Onuegbe	1985-01-01
	111035	Przemyslaw	Kaelbling	1985-01-01
	111400	Arie	Staelin	1985-01-01
	111692	Tonny	Butterworth	1985-01-01



Poizvedbe po več tabelah...

- Poizvedbe po več tabelah lahko izvajamo z uporabo **vgnezdenih SELECT stavkov**
- **Omejitev**: stolpci v rezultatu so lahko le iz ene tabele
- V poizvedbah, ki vračajo stolpce različnih tabel, moramo uporabljati **stik**.
 - v sklopu **FROM** navedemo tabele, v sklopu **WHERE** določimo stolpce za stik.



Poizvedbe po več tabelah...

- Za ločevanje med istoimenskimi stolpci uporabljamo **sinonime** (*alias*).

```
select E.emp_no, E.first_name, E.last_name  
from employees E, dept_manager DM, departments D  
where  
    E.emp_no = DM.emp_no AND  
    DM.dept_no = D.dept_no AND  
    D.dept_name = 'marketing';
```

emp_no	first_name	last_name
110022	Margareta	Markovitch
110039	Vishwani	Minakawa



Alternativni načini stika več tabel

- Alternativni načini stika med več tabelami:
 - FROM employees E JOIN titles T ON E.emp_no = T.emp_no
 - FROM employees JOIN titles USING emp_no
 - FROM employees NATURAL JOIN titles
- Zgornji zapisi nadomestijo sklopa FROM in WHERE
- V prvem primeru rezultat vsebuje dva identična stolpca .



Primer naravnega stika

```
select * from employees natural join titles;
```

emp_no	birth_date	first_name	last_name	gender	hire_date	title	from_date	to_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26	Senior Engineer	1986-06-26	9999-01-01
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	Staff	1996-08-03	9999-01-01
10003	1959-12-03	Parto	Bamford	M	1986-08-28	Senior Engineer	1995-12-03	9999-01-01
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01	Engineer	1986-12-01	1995-12-01
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01	Senior Engineer	1995-12-01	9999-01-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12	Senior Staff	1996-09-12	9999-01-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12	Staff	1989-09-12	1996-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02	Senior Engineer	1990-08-05	9999-01-01
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10	Senior Staff	1996-02-11	9999-01-01
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10	Staff	1989-02-10	1996-02-11
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15	Assistant Engi...	1998-03-11	2000-07-31
10009	1952-04-19	Sumant	Peac	F	1985-02-18	Assistant Engi...	1985-02-18	1990-02-18
10009	1952-04-19	Sumant	Peac	F	1985-02-18	Engineer	1990-02-18	1995-02-18
10009	1952-04-19	Sumant	Peac	F	1985-02-18	Senior Engineer	1995-02-18	9999-01-01
10010	1963-06-01	Diana	Burstein	F	1989-08-24	Engineer	1996-11-24	9999-01-01



Zunanji stik

■ Primer LEFT JOIN

```
select E.emp_no, E.first_name, E.last_name  
from employees E left join dept_manager DM  
on E.emp_no = DM.emp_no;
```

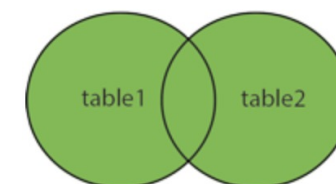
emp_no	first_name	last_name
10001	Georgi	Facello
10002	Bezalel	Simmel
10003	Parto	Bamford
10004	Chirstian	Koblick
10005	Kyoichi	Maliniak
10006	Anneke	Preusig
10007	Tzvetan	Zielinski
10008	Saniya	Kalloufi
10009	Sumant	Peac
10010	Duanakaew	Piveteau

Dodatne možnosti:

RIGHT JOIN

FULL OUTER JOIN

FULL OUTER JOIN





Uporaba operacij nad množicami...

- Rezultate dveh ali več poizvedb lahko združujemo z ukazi:
 - **UNION** (unija),
 - **INTERSECTION** (Presek)
 - **DIFFERENCE** (**EXCEPT**) (Razlika)
- Da lahko izvajamo našete operacije, morata tabeli A in B biti **skladni** (domene atributov morajo biti enake).
 - Različne implementacije imajo glede tega zelo različna pravila!



Uporaba operacij nad množicami

- Sintaksa:

```
op [ALL] [CORRESPONDING [BY  
{column1 [, ...]}]]
```

- Če uporabimo **CORRESPONDING BY**, se operacija izvede samo nad poimenovanimi stolpci
- Če uporabimo samo **CORRESPONDING** brez **BY** člena, se operacija izvede nad skupnimi stolpci.
- Če uporabimo **ALL**, lahko rezultat vključuje tudi dvojnike



V MySQL opcija **CORRESPONDING BY** ni na voljo.



Primer unije

- Izpiši prve tri zaposlene moške in prve tri zaposlene ženske.

```
select EM.first_name, EM.last_name, 'Moški'
from
    (select * from employees where gender = 'M'
     order by hire_date desc limit 3) EM
union
select EF.first_name, EF.last_name, 'Ženske'
from
    (select * from employees where gender = 'F'
     order by hire_date desc limit 3) EF;
```



INSERT stavek...

```
INSERT INTO TableName [ (columnList) ]  
VALUES (dataValueList)
```

- Seznam `columnList` ni obvezen;
- Pri vnosu moramo vpisati najmanj vse obvezne vrednosti (`not null`), razen za stolpce s privzeto vrednostjo (`DEFAULT`).
- Seznam `dataValueList` mora ustrezati seznamu `columnList`.



Primeri INSERT stavkov...

- Vnos nove vrstice v tabelo **departments**

```
insert into departments (dept_no, dept_name)
values ('d010', 'Education');
```

neobvezno

Shema relacije:
departments (dept_no, dept_name)

- Posebnosti:
 - Obravnava neobveznih polj
 - Privzete vrednosti



Primeri INSERT stavkov...

- Vnos več vrstic iz ene ali več drugih tabel...

```
INSERT INTO TableName [ (columnList) ]  
SELECT ...
```



UPDATE stavek...

UPDATE TableName

SET columnName1 = dataValue1

[, columnName2 = dataValue2...]

[WHERE searchCondition]

- **TableName** se lahko nanaša na ime osnovne tabele ali ime pogleda.
- Sklop **SET** določa nazive enega ali več stolpcev ter nove vrednosti teh stolpcev (morajo ustrezati po podatkovnem tipu).
- **WHERE** sklop neobvezen.



Primeri UPDATE stavkov

- Vsem zaposlenim dvigni plačo za 10%.

```
update salaries set salary = salary * 1.1  
where to_date = '9999-01-01'
```




DELETE stavek

```
DELETE FROM TableName  
[WHERE searchCondition]
```

- **TableName** se lahko nanaša na ime osnovne tabele ali ime pogleda.
- **WHERE sklop** ni obvezen. Če ga spustimo, zberemo vse vrstice v tabeli. Tabela ostane.



Primeri DELETE stavkov

- Izbriši podatke o plačah delavcev, ki niso več zaposleni.

```
delete from salaries
where emp_no not in (
    select emp_no
    from salaries
    where to_date = '9999-01-01'
);
```

Drop vs Truncate



Stavki skupine SQL DDL...

- **DDL skupina** zajema SQL stavke za manipulacijo s strukturo podatkovne baze.
- Kaj si bomo pogledali:
 - Podatkovni tipi, ki jih podpira SQL standard.
 - Mehanizmi za zagotavljanje kakovosti podatkov.
 - Uporaba mehanizmov za zagotavljanje kakovosti v **CREATE** in **ALTER TABLE** stavkih.
 - Način delovanja ISO transakcijskega modela.
 - Sprožilci, shranjene procedure in funkcije.

Podatkovni tipi v SQL standardu

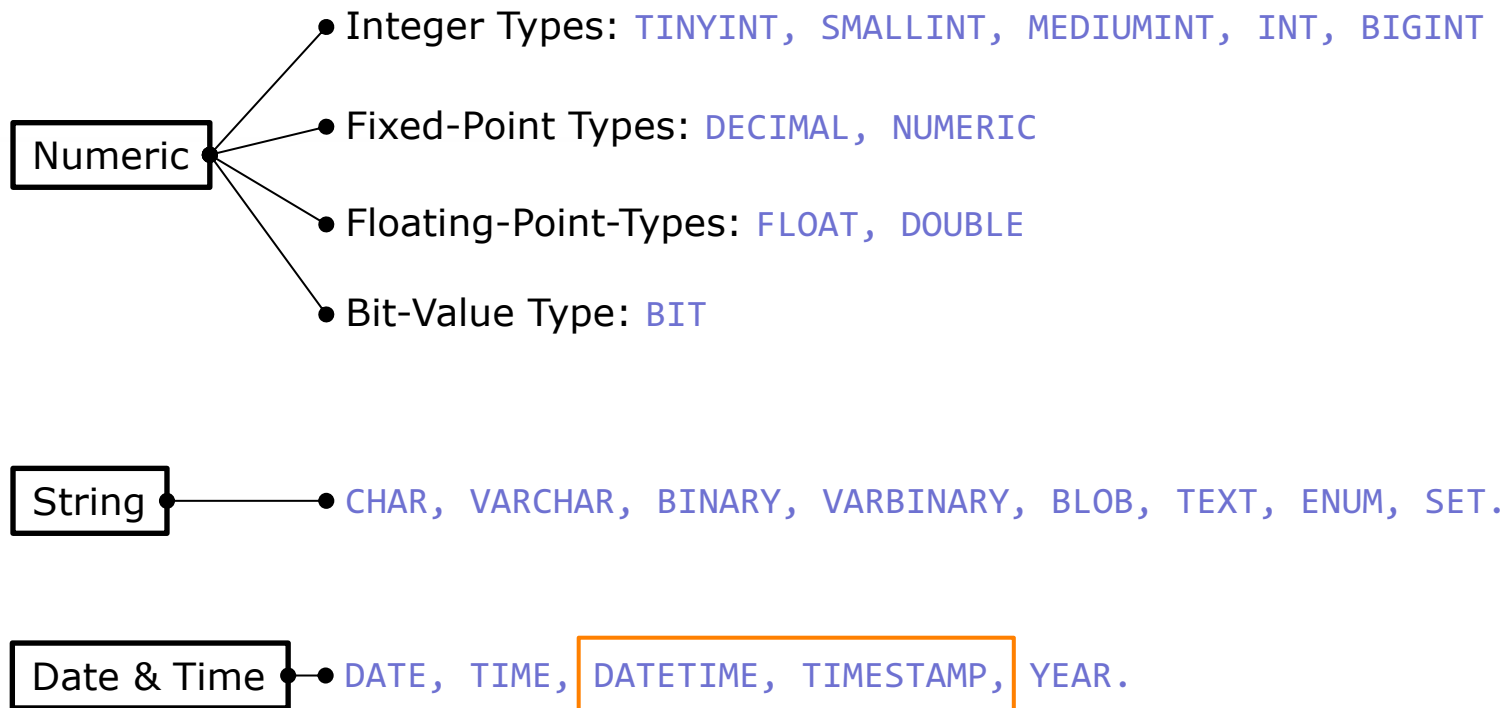
Table 6.1 ISO SQL data types.

Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	
interval	INTERVAL			
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT	

Vir: [2,159]



Podatkovni tipi v MySQL





Zagotavljanje kakovosti podatkov...

- SQL standard ponuja več vrst omejitev (*Integrity Enhancement Fetures*):
 - Obveznost podatkov
 - Omejitve domene (*Domain constraints*)
 - Pravila za celovitost podatkov (*Integrity constraints*)
 - Celovitost entitet (*Entity Integrity*)
 - Celovitost povezav (*Referential Integrity*)
 - Števnost (*Multiplicity*)
 - Splošne omejitve (*General constraints*)
- Omejitve so lahko definirane v **CREATE** in **ALTER TABLE** stavkih.



Zagotavljanje kakovosti podatkov...

- Obveznost podatkov

`emp_no numeric(5) NOT NULL`

- Omejitve domene

`gender CHAR NOT NULL`

`CHECK (gender in ('M', 'F'))`



Zagotavljanje kakovosti podatkov...

- Kreiranje domene

```
CREATE DOMAIN DomainName [AS] dataType  
[DEFAULT defaultOption]  
[CHECK (searchCondition)]
```

Primer:

```
CREATE DOMAIN Tgender AS CHAR  
CHECK (VALUE IN ('M', 'Ž'));
```

```
gender Tgender NOT NULL
```



MySQL ne podpira uporabniških domen.



Zagotavljanje kakovosti podatkov...

- `searchCondition` lahko vsebuje **iskalno tabelo**:

```
CREATE DOMAIN TempID AS numeric(5)  
CHECK (VALUE IN (SELECT emp_no FROM employees));
```

- Domeno lahko ukinemo z uporabo stavka `DROP DOMAIN`:

```
DROP DOMAIN DomainName  
[RESTRICT | CASCADE]
```

Pove, kako ravnati, če je
domena trenutno v uporabi



Celovitost entitet

- ISO standard podpira kreiranje primarnih in tujih ključev v okviru `CREATE` in `ALTER TABLE` stavkov.

```
PRIMARY KEY(dept_no, emp_no)
```

```
FOREIGN KEY(emp_no) REFERENCES employees
```

- Določimo lahko enolične stolpce ali kombinacije stolpcev.

```
UNIQUE(last_name)
```

```
UNIQUE(last_name, first_name)
```



Celovitost povezav...

- **Celovitost povezav:** če ima FK neko vrednost, potem se mora ta vrednost nahajati v primarnem ključu povezane tabele!

salaries

emp_no	salary	from_date	to_date
10001	60117	1986-06-26	1987-06-26
10001	62102	1987-06-26	1988-06-25
10001	66074	1988-06-25	1989-06-25
10001	66596	1989-06-25	1990-06-25
10001	66961	1990-06-25	1991-06-25
10001	71046	1991-06-25	1992-06-24

employees

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Christian	Koblick	M	1986-12-01
10005	1955-01-21	Kuznetsov	Malinin	M	1986-09-12



Celovitost povezav...

- Vsak **INSERT/UPDATE** stavek, ki skuša kreirati FK vrednost v tabeli, brez da bi ta vrednost obstajala kot PK v povezani tabeli, je zavrnjen.
- Ob zavrnitvi so možne naslednje akcije
 - **CASCADE**
 - **SET NULL**
 - **SET DEFAULT**
 - **NO ACTION**



MySQL opcije: **RESTRICT**, **CASCADE**, **SET NULL**, **NO ACTION**



Celovitost povezav...

- Določimo z uporabo `ON UPDATE`, `ON DELETE`,
`ON UPDATE SET NULL`

- Primeri:

```
FOREIGN KEY (emp_no) REFERENCES employees  
ON DELETE CASCADE
```

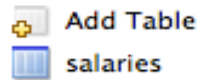
```
show create table salaries
```

Physical Schemata



employees
MySQL Schema

Tables (6 items)



Add Table



salaries



departments



titles



dept_emp



dept_manager



employees

Views (0 items)



Add View

Routines (0 items)



Add Routine

Routine Groups (0 items)



Add Group

Schema Privileges

SQL Scripts

Model Notes

salaries - Table



Name: salaries

Schema: employees

Foreign Key | Referenced Table

salaries_ibfk_1 | `employees`.`employees`

<click to edit>

Foreign key details 'salaries_ibfk_1'

Column | Referenced Column

☒ emp_no | emp_no

☐ salary

☐ from_date

☐ to_date

On Update: RESTRICT

On Delete: CASCADE

Comment:

☐ Skip on SQL generation



IEF – Splošne omejitve

- Splošne omejitve določimo z **CHECK/UNIQUE** opcijami v **CREATE** in **ALTER TABLE** stavkih.

```
CREATE ASSERTION AssertionName  
CHECK (searchCondition)
```



MySQL ne podpira splošnih omejitev. V večini primerov zadoščajo sprožilci (**trigger**)



Primer splošne omejitve

- Poslovno pravilo: nobenemu zaposlenemu se plača ne sme v istem letu povečati več kot trikrat.

```
CREATE ASSERTION PrevecSprememb  
CHECK not exists (  
    select count(*)  
    from salaries  
    group by emp_id, YEAR(date_from)  
    having count(*) > 3  
);
```

Ali ta SQL stavek
ustreza našim
zahtevam?



Kreiranje podatkovnih objektov

- SQL DDL omogoča **kreiranje in brisanje podatkovnih objektov**, kot so: shema, domena, tabela, pogled in indeks.
- Glavni SQL DDL stavki:

CREATE SCHEMA

DROP SCHEMA

CREATE/ALTER DOMAIN

DROP DOMAIN

CREATE/ALTER TABLE

DROP TABLE

CREATE/ALTER VIEW

DROP VIEW

CREATE INDEX

DROP INDEX

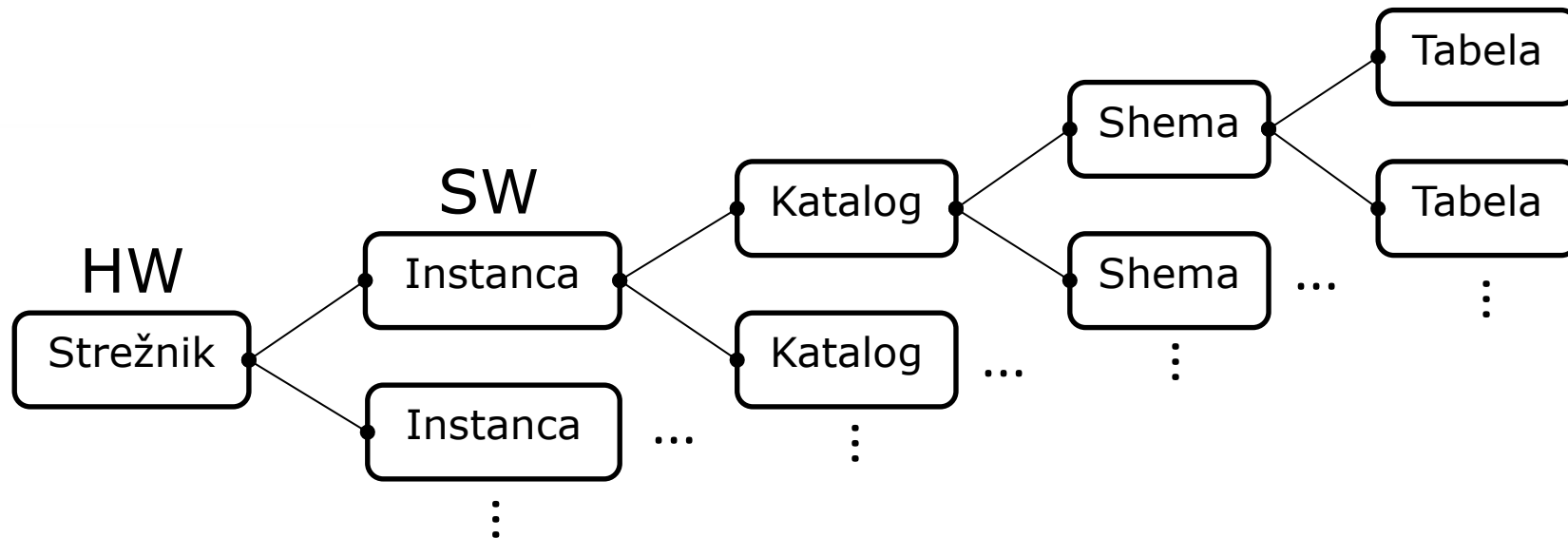


Kreiranje podatkovnih objektov...

- Relacije in drugi podatkovni objekti obstajajo v nekem okolju.
 - Vsako okolje vsebuje enega ali več katalogov, vsak **katalog** pa množico shem.
 - Shema je poimenovana **kolekcija povezanih podatkovnih objektov**.
 - Objekti v shemi so lahko **tabele**, **pogledi**, **domene**, **trditve**, **dodelitve**, **pretvorbe** in **znakovni nizi**. Vsi objekti imajo istega lastnika.
- Med implemantacijami razlike v poimenovanju.

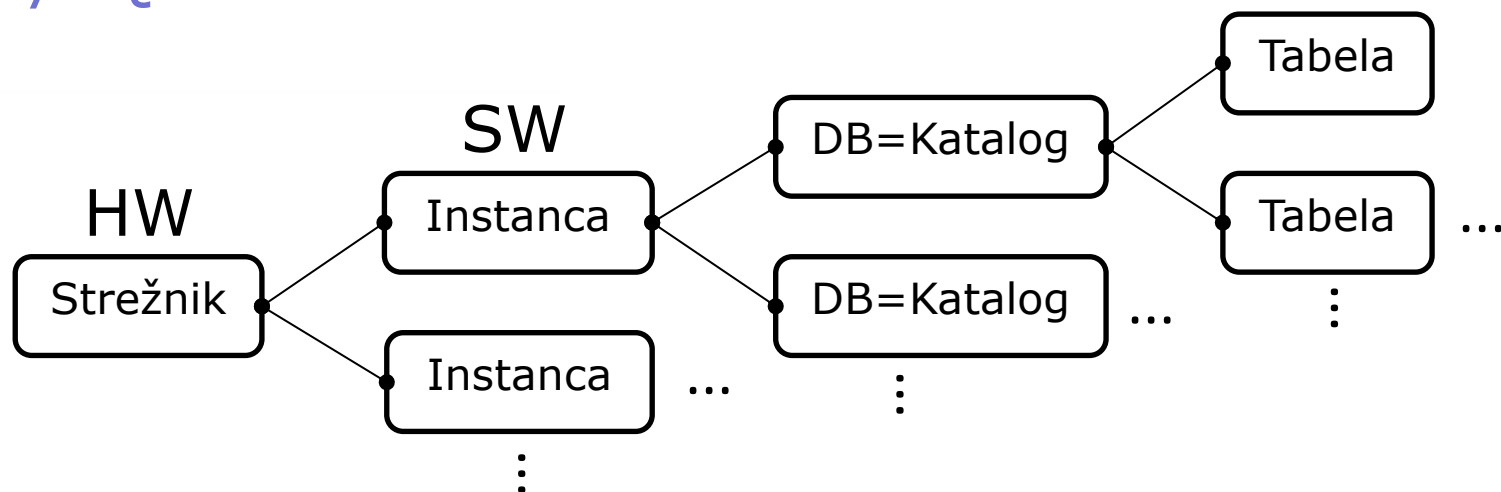


Kreiranje podatkovnih objektov...





Kreiranje podatkovnih objektov...





Kreiranje sheme

```
CREATE SCHEMA [Name |  
    AUTHORIZATION CreatorId ]
```

```
DROP SCHEMA Name [RESTRICT | CASCADE]
```

- **RESTRICT** (privzeto): shema mora biti prazna, sicer brisanje ni možno.
- **CASCADE**: kaskadno se brišejo vsi objekti, povezani s shemo. Če katerokoli brisanje ne uspe, se zavrne celotna operacija.



Kreiranje tabele...

```
CREATE TABLE TableName  
({colName dataType [NOT NULL] [UNIQUE]  
[DEFAULT defaultOption]  
[CHECK searchCondition] [,...]}  
[PRIMARY KEY (listOfColumns),]  
{[UNIQUE (listOfColumns),] [...,]}  
{[FOREIGN KEY (listOfFKColumns)  
REFERENCES ParentTableName [(listOfCKColumns)],  
[ON UPDATE referentialAction]  
[ON DELETE referentialAction ]] [,...]}  
{[CHECK (searchCondition)
```



Primer kreiranja tabele



Primer CREATE stavka v MySQL

```
CREATE TABLE dept_manager (  
    dept_no CHAR(4) NOT NULL,  
    emp_no INT(11) NOT NULL,  
    from_date DATE NOT NULL,  
    to_date DATE NOT NULL,  
    PRIMARY KEY (emp_no, dept_no),  
    CONSTRAINT dept_manager_ibfk_1  
        FOREIGN KEY (emp_no)  
        REFERENCES employees(emp_no)  
        ON DELETE CASCADE,  
    CONSTRAINT dept_manager_ibfk_2  
        FOREIGN KEY (dept_no)  
        REFERENCES departments (dept_no)  
        ON DELETE RESTRICT)
```



Vaja

- Kreirati želimo podatkovno bazo za hranjenje filmov – Filmoteka.
- Tabela FILM
 - Naslov
 - Avtorji
 - Žanr
 - Ocena IMDB
 - Moja ocena
 - Tip
 - Jezik
 - Videnost



Kreiramo in izberemo podatkovno bazo

```
create database filmoteka;  
use filmoteka;
```



Tabele

- Ali poleg tabele filmoteka potrebujemo še kakšno tabelo? So atributi vsi enostavni?

Naslov

Avtorji

Žanr

Ocena IMDB

Moja ocena

Tip

Jezik

Videno



Kreiramo tabelo avtor

```
create table avtor (  
    avtor_id int(4),  
    ime char(20) not null,  
    priimek char(20) not null,  
    primary key (avtor_id));
```

SCHEMAS

Search objects

- ▶ Citations
- ▶ employees
- ▼ **filmoteka**
 - ▼ Tables
 - ▼ **avtor**
 - ▼ Columns
 - ◆ avtor_id
 - ◆ ime
 - ◆ priimek
 - ▼ Indexes
 - ▶ PRIMARY
 - ▶ Foreign Keys
 - ▶ Triggers
 - Views
 - Routines

Query 1

```
1 use filmoteka;
2 create table avtor (
3     avtor_id int(4),
4     ime char(20) not null,
5     priimek char(20) not null,
6     primary key (avtor_id));
7
```



Kreiramo ostale povezane tabele

```
create table zanr (  
    zanr_id int(4),  
    naziv char(50) not null,  
primary key (zanr_id));
```

```
create table tip (  
    tip_id int(4),  
    naziv char(50) not null,  
primary key (tip_id));
```

```
create table jezik (  
    jezik_id int(4),  
    naziv char(50) not null,  
primary key (jezik_id));
```

The screenshot shows a database management interface. On the left is a 'SCHEMAS' panel with a search bar and a tree view. The tree view shows a hierarchy: 'Citations', 'employees', 'filmoteka' (expanded), 'Tables' (expanded), and then a list of tables: 'avtor', 'jezik', 'tip', and 'zanr'. Below the tables are 'Views' and 'Routines'. Other databases listed include 'HotelChain', 'Library', 'Movies', 'mydb', 'northwind', 'sakila', 'sys', 'test', 'URARNA', 'watchstore', and 'wikileaks'. The main area on the right is a 'Query 1' editor. It contains SQL code to create a database and four tables. The code is as follows:

```
1 create database filmoteka;
2
3 use filmoteka;
4
5 create table avtor (
6     avtor_id int(4),
7     ime char(20) not null,
8     priimek char(20) not null,
9     primary key (avtor_id));
10
11 create table zanr (
12     zanr_id int(4),
13     naziv char(50) not null,
14     primary key (zanr_id));
15
16 create table tip (
17     tip_id int(4),
18     naziv char(50) not null,
19     primary key (tip_id));
20
21 create table jezik (
22     jezik_id int(4),
23     naziv char(50) not null,
24     primary key (jezik_id));
25
```



Kreiramo tabelo film

```
create table film (  
    film_id int(6),  
    naslov char(50),  
    avtor int(6),  
    zanr int(4),  
    ocenaIMDB decimal(4,1),  
    mojaOcena decimal(4,1),  
    tip int(4),  
    jezik int(4),  
    videno bit,  
    primary key (film_id),  
    ...
```



...povezave na druge tabele

...

```
constraint fk_avtor foreign key (avtor)
  references avtor(avtor_id) on delete restrict
constraint fk_tip foreign key (tip)
  references tip(tip_id) on delete cascade,
constraint fk_jezik foreign key (jezik)
  references jezik(jezik_id) on delete cascade
  on update cascade,
constraint fk_zanr foreign key (zanr)
  references zanr(zanr_id) on delete cascade on
  update restrict
);
```


SQL

SQL

SCHEMAS

Search objects

Citations

employees

filmoteka

Tables

avtor

film

Columns

film_ID

naslov

zanr

ocenaIMDB

mojaOcena

tip

jezik

videno

Indexes

Foreign Keys

fk_jezik

fk_tip

fk_zanr

Query 1

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

```

create table film (
    film_ID int(6),
    naslov char(50),
    zanr int(4),
    ocenaIMDB decimal(4,1),
    mojaOcena decimal(4,1),
    tip int(4),
    jezik int(4),
    videno bit,
    primary key (film_id),
    constraint fk_tip foreign key (tip)
        references tip(tip_id) on delete cascade,
    constraint fk_jezik foreign key (jezik)
        references jezik(jezik_id) on delete cascade,
    constraint fk_zanr foreign key (zanr)
        references zanr(zanr_id) on delete cascade
);

```

100%

6:1

Action Output

	Time	Action
✓ 1	09:45:23	create table avtor (avtor_id int(4), ime char(20) not null, priimek char(20) not null, primary key (avtor_id))
✓ 2	09:45:38	use filmoteka
✓ 3	09:45:41	create table avtor (avtor_id int(4), ime char(20) not null, priimek char(20) not null, primary key (avtor_id))
✓ 4	09:49:38	create table jezik (jezik_id int(4), naziv char(50) not null, primary key (jezik_id))

Object Info

Session

Foreign Key: **fk_jezik**

Definition:

jezik (jezik → jezik_id)

99

Predmet: OPB, Modul: Poizvedovalni jezik SQL, Gradivo: v.2017, ©UL FRI, Laboratorij za podatkovne tehnologije



Obveznost povezav

- Vnesimo nekaj podatkov:

```
insert into avtor (avtor_id, ime, priimek) values (...)  
insert into zanr (zanr_id, naziv) values (...)  
insert into tip (tip_id, naziv) values (...)  
insert into jezik (jezik_id, naziv) values (...)  
insert into film (film_id, naslov, avtor, zanr,  
                ocenaIMDB, mojaOcena, tip, jezik, videno) values (...)
```

```
insert into avtor (avtor_id, ime, priimek) values (1, 'David', 'Fincher'), (2, 'Matevž', 'Luzar');  
insert into zanr (zanr_id, naziv) values (1, 'Kriminalka'), (2, 'Akcija'), (3, 'Drama');  
insert into tip (tip_id, naziv) values (1, 'Film'), (2, 'TV serija'), (3, 'Risani film');  
insert into jezik (jezik_id, naziv) values (1, 'Angleški'), (2, 'Nemški'), (3, 'Slovenski');  
insert into film (film_id, naslov, avtor, zanr, ocenaIMDB, mojaOcena, tip, jezik, videno)  
values (1, 'Se7en', 1, 1, 8.6, 9, 1, 1, 1), (1, 'Jezero', 2, 1, 7.9, 8.2, 2, 3, 1);
```

Kaj pa podatki o igralcih v filmu?



ALTER TABLE stavek...

- S stavkom **ALTER TABLE** lahko:
 - Dodajamo ali ukinjamo stolpce v tabeli;
 - Dodajamo ali ukinemo omejitve tabele;
 - Za stolpce v tabeli določamo ali ukinjamo privzete vrednosti;
 - Spreminjamo podatkovne tipe stolpcev v tabeli;



Primeri ALTER TABLE stavkov...

- Spremeni tabelo `dept_manager` tako, da ukineš privzeto vrednost stolpca `from_date`.

```
ALTER TABLE dept_manager  
  ALTER from_date DROP DEFAULT;
```



Primeri ALTER TABLE stavkov...

- Spremeni tabelo **salaries** tako, da ukineš omejitev **PrevecSprememb**. Tabeli **employees** dodaj stolpec **retired** s privzeto vrednost 'N'.

```
ALTER TABLE salaries  
DROP CONSTRAINT PrevecSprememb;
```

```
ALTER TABLE employees  
ADD retired CHAR(1) NOT NULL DEFAULT 'N';
```



Primeri ALTER TABLE

```
ALTER TABLE employees ADD nickname AFTER first_name;
```

```
ALTER TABLE employees CHANGE first_name fn;
```

```
ALTER TABLE employees MODIFY birth_date date NULL;
```

```
ALTER TABLE employees ALTER gender SET DEFAULT 'M';
```

```
ALTER TABLE employees DROP gender;
```

```
DESCRIBE employees;
```



Stavek DROP TABLE

- S pomočjo stavka **DROP TABLE** ukinemo tabelo. Obenem se zbrisejo vsi zapisi tabele.

```
DROP TABLE TableName [RESTRICT | CASCADE]
```

- Primer:

```
DROP TABLE employees RESTRICT;
```

Glej še: **TRUNCATE**, **CREATE LIKE** + **DROP**



Indeksi

- Omogočajo urejanje tabel po različnih stolpcih. So ključnega pomena za hitro poizvedovanje.
- Splošna sintaksa:

```
CREATE [UNIQUE] INDEX index_name  
ON table_name  
    (column1 [ASC|DESC],  
     column2 [ASC|DESC],  
     ...);
```



Indeksi

- Sintaksa v MySQL: 

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name  
[USING {BTREE|HASH}] ON tbl_name (index_col_name,...)  
[index_option][algorithm_option|lock_option]
```

index_col_name: col_name [(length)] [ASC|DESC]

index_option:

```
KEY_BLOCK_SIZE[=]value|  
USING {BTREE|HASH}|  
WITH PARSER parser_name|  
COMMENT 'string'
```

algorithm_option: ALGORITHM[=]{DEFAULT|INPLACE|COPY}

lock_option: LOCK[=]{DEFAULT|NONE|SHARED|EXCLUSIVE}



Primer kreiranja indeksa

- Kreiraj indeks na tabeli **Employees**, in sicer nad stolpci ime, priimek in spol. Indeks naj bo urejen naraščajoče po imenu in padajoče po priimku.

```
CREATE INDEX idx_emp_fullname_gender  
ON employees  
(first_name ASC, last_name DESC, gender);
```

```
SHOW INDEX FROM employees;
```



Pogledi

- Predstavljajo navidezne tabele. Vsakokrat, ko dostopamo do pogleda, se v ozadju izvede SELECT stavek.
- Splošna sintaksa

```
CREATE VIEW view_name [(column_list)]  
AS select_statement  
[WITH [CASCADED | LOCAL] CHECK OPTION]
```



Primer kreiranja pogleda

- Kreiraj pogled, ki prikazoval podatke o tem, kdo so trenutno vodje posameznih oddelkov ter od kdaj so na funkciji.

```
CREATE VIEW Managers AS
    SELECT e.emp_no, e.first_name, e.last_name,
           d.dept_name, dm.from_date
    FROM employees e
    JOIN dept_manager dm ON e.emp_no = dm.emp_no
    JOIN departments d ON d.dept_no = dm.dept_no
    WHERE dm.to_date = '9999-01-01';
```



Transakcije...

- SQL definira **transakcijski model** z ukazoma **COMMIT** in **ROLLBACK**.
- **Transakcija**: logična enota dela z enim ali več SQL ukazi. Je **atomarna**.
- Spremembe znotraj transakcije **praviloma** drugim transakcijam skrite, dokler transakcija ni končana.



Transakcije...

- Transakcija se lahko zaključi **eksplicitno** (**COMMIT/ROLLBACK**) ali **implicitno** (skupaj s programom, v katerem klicana).
- Nova transakcija se začne z novim SQL stavkom, ki transakcijo inicira.
- SQL transakcij ne moremo gnezditi.
 - Pravilo ne velja pri vseh implementacijah.



Transakcije

- Transakcijo nastavimo s pomočjo ukaza `SET TRANSACTION`

`SET TRANSACTION`

`[READ ONLY | READ WRITE] |`

`[ISOLATION LEVEL READ UNCOMMITTED |`

`READ COMMITTED|REPEATABLE READ |SERIALIZABLE]`



Transakcije...

- **READ ONLY** – pove, da transakcija vključuje samo operacije, ki iz baze berejo.
 - SUPB bo dovolil **INSERT**, **UPDATE** in **DELETE** samo nad začasnimi tabelami.
- **ISOLATION LEVEL** – pove stopnjo interakcije, ki jo SUPB dovoli med to in drugimi transakcijami.
 - **READ UNCOMMITTED**
 - **READ COMMITTED**
 - **REPEATABLE READ**
 - **SERIALIZABLE**
- Varen je samo način **SERIALIZABLE** (vrača serializabilne urnike)



Izolacijske ravni transakcij

Dirty Read

users		
id	name	age
1	Joe	20
2	Jill	25

Transaction 1

```
/* Query 1 */  
SELECT age FROM users WHERE id = 1;  
/* will read 20 */
```

```
/* Query 1 */  
SELECT age FROM users WHERE id = 1;  
/* will read 21 */
```

Transaction 2

```
/* Query 2 */  
UPDATE users SET age = 21 WHERE id = 1;  
/* No commit here */
```

```
ROLLBACK; /* lock-based DIRTY READ */
```

Dirty Read se lahko zgodi pri izolaciji Read Uncommitted



Izolacijske ravni transakcij

Non-repeatable read

Transaction 1

```
/* Query 1 */  
SELECT * FROM users WHERE id = 1;
```

```
/* Query 1 */  
SELECT * FROM users WHERE id = 1;  
COMMIT; /* lock-based REPEATABLE READ */
```

Transaction 2

```
/* Query 2 */  
UPDATE users SET age = 21 WHERE id = 1;  
COMMIT; /* in multiversion concurrency  
control, or lock-based READ COMMITTED */
```

Non-repeatable read se lahko zgodi pri izolacijskih ravneh
Read Committed, Read Uncommitted



Izolacijske ravni transakcij

Phantom read

Transaction 1

```
/* Query 1 */  
SELECT * FROM users  
WHERE age BETWEEN 10 AND 30;
```

```
/* Query 1 */  
SELECT * FROM users  
WHERE age BETWEEN 10 AND 30;
```

Transaction 2

```
/* Query 2 */  
INSERT INTO users VALUES ( 3, 'Bob', 27 );  
COMMIT;
```

Phantom read se lahko zgodi pri vseh izolacijskih ravneh razen pri **Serializable**



Izolacijske ravni transakcij

Isolation level	Dirty reads	Non-repeatable reads	Phantoms
Read Uncommitted	●	●	●
Read Committed		●	●
Repeatable Read			●
Serializable			



Delo s transakcijami v MySQL

- Sintaksa:

```
SET TRANSACTION [GLOBAL|SESSION]
[READ ONLY | READ WRITE] |
[ISOLATION LEVEL READ UNCOMMITTED |
READ COMMITTED|REPEATABLE READ |SERIALIZABLE ]
```

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.02
```

```
mysql> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
ERROR 1568 (25001): Transaction characteristics can't be changed
while a transaction is in progress
```



Delo s transakcijami v MySQL

- Inicializacija transakcije

`START TRANSACTION [READ WRITE | READ ONLY]`

SQL stavki

Eksplicitno zaključevanje transakcije

- Eksplicitno zaključevanje transakcije:

`COMMIT | ROLLBACK`


- Samodejno zaključevanje transakcije:

`SET autocommit = {0 | 1}`



Transakcije v MySQL Workbench

- Seja v MySQL Workbench 
- Primer:



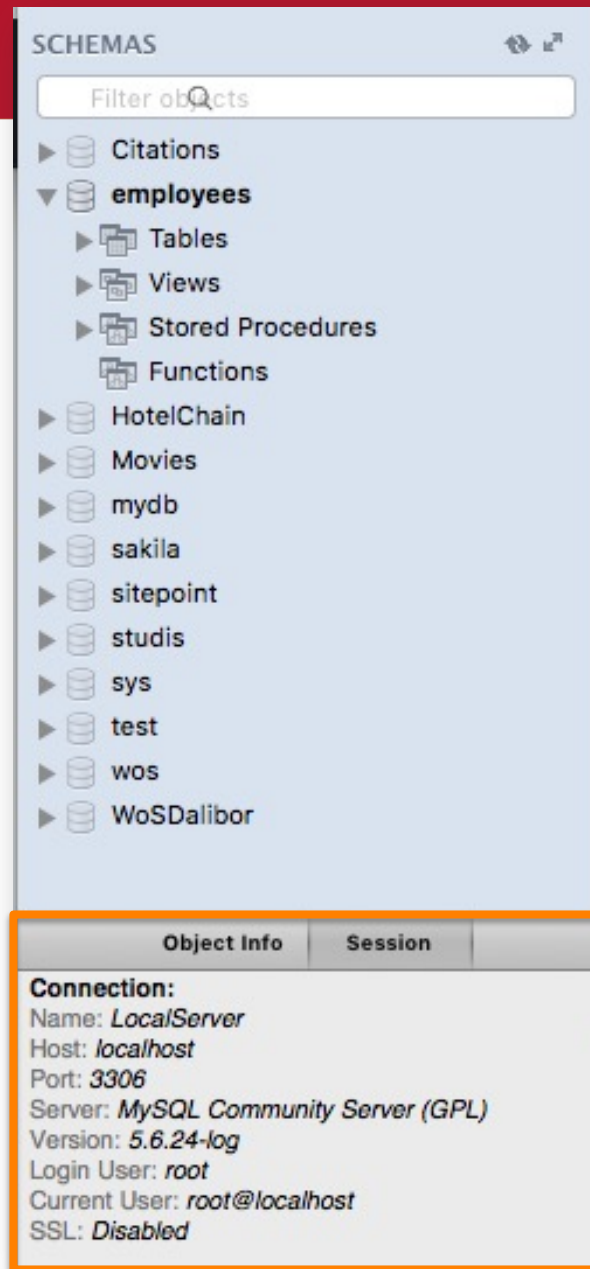
```
use employees;

select * from departments order by dept_no;

insert into departments (dept_no, dept_name)
values ('d011', 'Public relationships');

delete from departments where dept_no = 'd011';

start transaction;
insert into departments (dept_no, dept_name)
values ('d011', 'Public relationships');
rollback;
```



SCHEMAS

Filter objects

- ▶ Citations
- ▼ **employees**
 - ▶ Tables
 - ▶ Views
 - ▶ Stored Procedures
 - ▶ Functions
- ▶ HotelChain
- ▶ Movies
- ▶ mydb
- ▶ sakila
- ▶ sitepoint
- ▶ studis
- ▶ sys
- ▶ test
- ▶ wos
- ▶ WoSDalibor

Object Info	Session
Connection: Name: LocalServer Host: localhost Port: 3306 Server: MySQL Community Server (GPL) Version: 5.6.24-log Login User: root Current User: root@localhost SSL: Disabled	



Takojšnje in zapoznele omejitve...

- Upoštevanje omejitev: ob začetku/po zaključku transakcije.
 - **INITIALLY IMMEDIATE** – ob začetku transakcije;
 - **INITIALLY DEFERRED** – ob zaključku transakcije.

salaries

emp_no	salary	from_date	to_date
10001	60117	1986-06-26	1987-06-26
10001	62102	1987-06-26	1988-06-25
10001	66074	1988-06-25	1989-06-25
10001	66596	1989-06-25	1990-06-25
10001	66961	1990-06-25	1991-06-25
10001	71046	1991-06-25	1992-06-24

Omejitev: za vsakega delavca mora obstajati vsaj en zapis o plači.

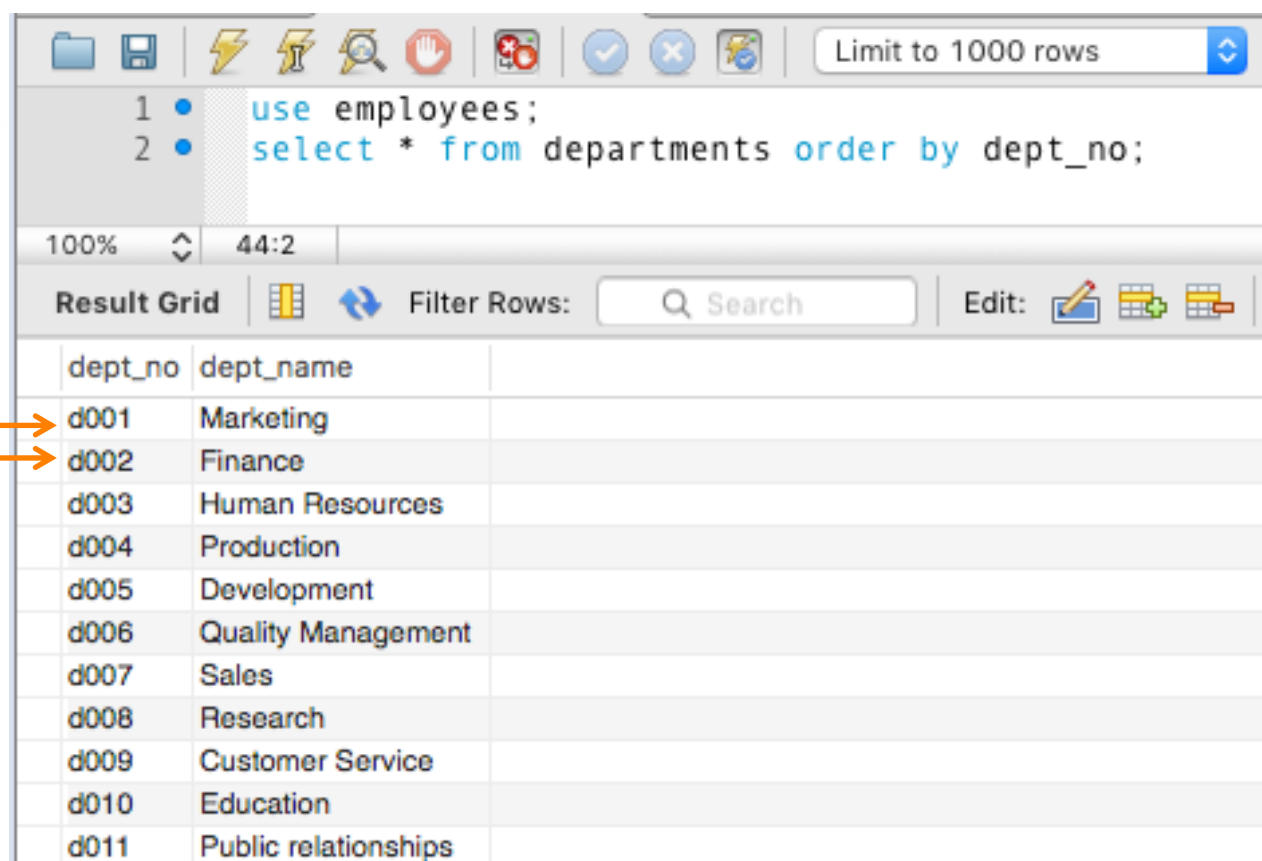
employees

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kuzochi	Malinski	M	1986-09-12



Takojšnje in zapoznele omejitve...

- Primer: zamenjava ključa



Limit to 1000 rows

```
1 • use employees;  
2 • select * from departments order by dept_no;
```

100% 44:2

Result Grid Filter Rows: Search Edit:

dept_no	dept_name
d001	Marketing
d002	Finance
d003	Human Resources
d004	Production
d005	Development
d006	Quality Management
d007	Sales
d008	Research
d009	Customer Service
d010	Education
d011	Public relationships



Takojšnje in zapoznele omejitve

- Način upoštevanja omejitev za trenutno transakcijo nastavimo z ukazom `SET CONSTRAINTS`.

SET CONSTRAINTS

{ALL | constraintName [, . . .]}

{DEFERRED | IMMEDIATE}



Takojšnje in zapoznele omejitve...

- Če izberemo **INITIALLY IMMEDIATE** (privzeta možnost), lahko določimo tudi, ali je zakasnitev moč določiti kasneje. Uporabimo **[NOT] DEFERRABLE**.

```
ALTER TABLE tab1 ADD CONSTRAINT fk_tab1_tab2
    FOREIGN KEY (tab2_id)
    REFERENCES tab2(id)
    DEFERRABLE
    INITIALLY IMMEDIATE;
```

```
ALTER SESSION SET CONSTRAINTS = DEFERRED;
ALTER SESSION SET CONSTRAINTS = IMMEDIATE;
```



MySQL ne podpira zapozneli omejitev...



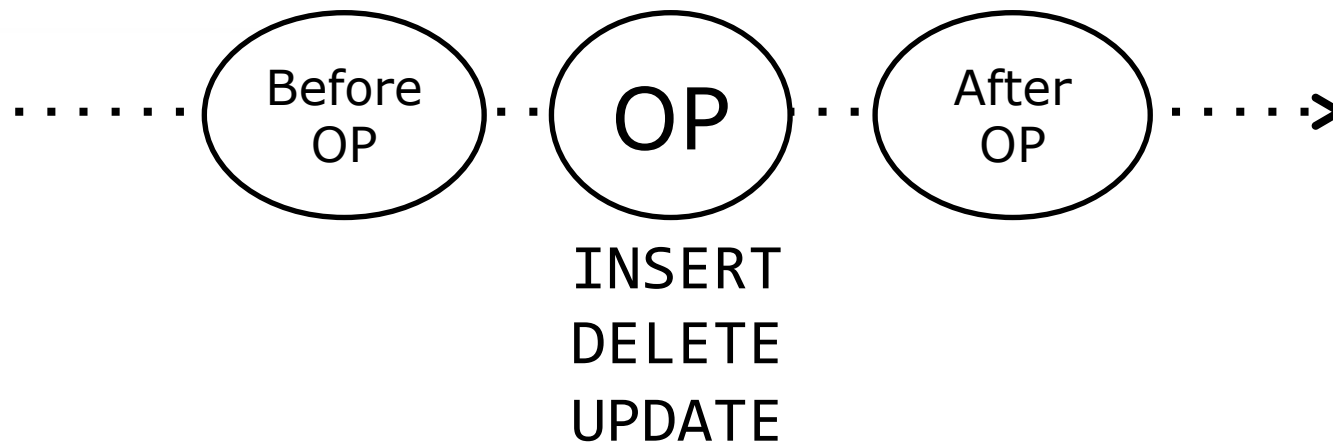
Bazni izvedbeni objekti

- Večina implementaciji SUPB podpira:
 - Sprožilce (*Trigger*)
 - V bazi shranjene **podprograme** (*Stored procedure*)
 - Funkcije (*Functions*)
- PROS:
 - Hitro izvajanje, izvedbeni objekti že prevedeni!
- CONS:
 - vključevanje poslovne logike na podatkovno raven.



Sprožilci...

- Se prožijo pred ali po opearciji INS, DEL, UPD





Sprožilci...



- Sintaksa v MySQL:

```
CREATE TRIGGER trigger_name  
trigger_time  
trigger_event  
ON tbl_name FOR EACH ROW  
[trigger_order] trigger_body;
```

```
trigger_time = {BEFORE, AFTER}
```

```
trigger_event = {INSERT, UPDATE, DELETE}
```

```
Trigger_order = {FOLLOWS | PRECEDES} other_trigger_name
```



Sprožilci

- Primer sprožilca:
 - V tabelo **employees** dodamo polje **avgSalary**, ki ga spremenimo vsakokrat, ko se delavcu spremeni plača.

```
alter table employees
add column avgSalary decimal(10,2) null;

create trigger upd_avgSalary
after insert on salaries
for each row update employees set avgSalary = (
    select avg(salary) from salaries
    where emp_no = NEW.emp_no)
where emp_no = NEW.emp_no;
```




Shranjeni podprogrami...



- Sintaksa v MySQL

```
CREATE PROCEDURE sp_name  
    ([proc_parameter[,...]])  
    [characteristic ...] routine_body
```

```
CREATE FUNCTION sp_name  
    ([func_parameter[,...]])  
    RETURNS type  
    [characteristic ...] routine_body
```

```
Func_parameter, proc_parameter:  
    [ IN | OUT | INOUT ] param_name type
```



Shranjeni podprogrami...



- Sintaksa v MySQL

characteristic:

```
COMMENT 'string' |  
LANGUAGE SQL |  
[NOT] DETERMINISTIC |  
{CONTAINS SQL|NO SQL|READS SQL DATA|MODIFIES SQL DATA}|  
SQL SECURITY { DEFINER | INVOKER }
```



Shranjeni programi



- Primer shranjene funkcije in njenega klica v sintaksi MySQL:

```
create function hello (empno INT(5))  
  returns char(50) deterministic  
  return concat('Pozdravljen ',  
    (select first_name  
     from employees  
     where emp_no = empno));
```

```
mysql> SELECT hello(10001);
```

```
mysql> Pozdravljen Georgi
```



Shranjeni programi



- Primer shranjene procedure in njenega klica v sintaksi mySQL:

```
create procedure DoubleIt (INOUT p1 INT)
begin
    select p1*p1 into p1;
end;
```

```
mysql> set @a = 10;
mysql> call DoubleIt(@a);
mysql> select @a;
mysql> 100
```

```
select DoubleIt(@a);
ERROR 1305 (42000):
FUNCTION employees.DoubleIt
does not exist
```



Vaja

- Izpiši naslednje podatke o zaposlenih:
 - Ime
 - Priimek
 - Starost
- Če gre za žensko, ne izpiši dejanske starosti, temveč starost v obliki:
 - Če starost manjša od 60 let \Rightarrow 'Pod 60', sicer 'Nad 60';

	first_name	last_name	Starost
►	Georgi	Facello	64
	Bezalel	Simmel	Pod 60
	Parto	Bamford	58
	Chirstian	Koblick	63
	Kyoichi	Maliniak	62
	Anneke	Preusig	Nad 60
	Tzvetan	Zielinski	Pod 60
	Saniya	Kalloufi	59
	Sumant	Peac	Nad 60
	Duangkaew	Piveteau	Pod 60
	Mary	Sluis	Nad 60
	Patricio	Bridgland	57



Vaja

```
create function empAge (empno int(5))
returns int(3) not deterministic reads sql data
begin
    return (select YEAR(now()) - YEAR(birth_date)
    from employees where emp_no = empno);
end
//delimiter ;
```

#Primer uporabe funkcije

```
select e.first_name, e.last_name,
if(gender='F',(if (empAge(e.emp_no)>60, 'Nad 60', 'Pod
60')), empAge(e.emp_no)) as 'Starost' from employees e;
```

IF(condition, A, B);

Če je pogoj izpolnjen, vrni A sicer B.



Dostopanje do MySQL iz programskih jezikov

- Primer v jeziku Java
- Primer v jeziku Ruby
- Primer v jeziku Python



Primer v jeziku Java



```
5 package dbtest;
6
7 import java.io.*;
8 import java.sql.*;
9 import java.util.*;
10
11 /**
12  *
13  * @author MarkoB
14  */
15 public class DBTest {
16
17     private static final String PROP_FILE = "db.properties";
18
19     //supporting method for making a connection
20     private static Connection getConnection() throws
21         IOException,
22         ClassNotFoundException,
23         InstantiationException,
24         IllegalAccessException,
25         SQLException {
26
27         //settings are read from a file
28         Properties p = new Properties();
29         p.load(new FileInputStream(PROP_FILE));
30         //dynamically loading the driver
31         String driver = p.getProperty("jdbc.driver");
32         Class.forName(driver).newInstance();
33
34         //opening connection to datasource
35         return DriverManager.getConnection(
36             p.getProperty("jdbc.conn"),
37             p.getProperty("jdbc.uid"),
38             p.getProperty("jdbc.pwd")
39         );
40     }
```




Vsebina datoteke db.properties

```
//gonilnik  
jdbc.driver=com.mysql.jdbc.Driver  
//baza, do katere želimo dostopati  
jdbc.conn=jdbc:mysql://localhost/citations  
//uporabniško ime  
jdbc.uid=root  
//geslo - če pustimo prazno, nas vpraša  
jdbc.pwd=
```

```

public static void main(String[] args) {
    try {
        //open connection
        System.out.println("odpiram povezavo");
        Connection dbconn = getConnection();
        System.out.println("povezava odprta");

        Statement s = dbconn.createStatement();
        System.out.println("INSERT stavek kreiran");

        //inserting a new row in table Author
        s.executeUpdate("insert into Author (AuthorID, AuthorName, "
            + "AuthorSurname) values (null, 'Marko', 'Bajec')");
        System.out.println("INSERT izveden");
        s.close();
        dbconn.close();
        System.out.println("povezava zaprta");

    } catch (SQLException e) {
        while (e != null) {
            e.printStackTrace();
            e = e.getNextException();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```



```
#!/usr/bin/ruby

require "mysql"

begin

  con = Mysql.new 'localhost', 'user12', '34klq*', 'mydb'

  rs = con.query("SELECT * FROM Writers")
  n_rows = rs.num_rows

  puts "There are #{n_rows} rows in the result set"

  n_rows.times do
    puts rs.fetch_row.join("\s")
  end

rescue Mysql::Error => e
  puts e.errno
  puts e.error

ensure
  con.close if con
end
```



```
import datetime
import mysql.connector

cnx = mysql.connector.connect(user='scott', database='employees')
cursor = cnx.cursor()

query = ("SELECT first_name, last_name, hire_date FROM employees "
        "WHERE hire_date BETWEEN %s AND %s")

hire_start = datetime.date(1999, 1, 1)
hire_end = datetime.date(1999, 12, 31)

cursor.execute(query, (hire_start, hire_end))

for (first_name, last_name, hire_date) in cursor:
    print("{} , {} was hired on {:%d %b %Y}".format(
        last_name, first_name, hire_date))

cursor.close()
cnx.close()
```



Koristne povezave

- MySQL SUPB
 - <http://www.mysql.com>
- employees DB
 - <https://launchpad.net/test-db/+download>
- SQL checker
 - <http://developer.mimer.com/validator/parser92/index.tml>
- Nekatera MySQL orodja
 - SQLeo: <http://sourceforge.net/projects/sqleo/>
 - Sequel Pro: <http://www.sequelpro.com>
 - Mac SQL studio: <http://macsqlstudio.com>