CSC7072: Databases, fall 2015

Dr. Kim Bauters

Domain Description Language (DDL)

Domain Description Language (DDL)

retrieving data using SQL

don't forget:

```
SELECT {attribute [AS new_attribute_name]}
FROM {table [AS new_table_name]}
[{JOIN table ON attribute = attribute}]
[WHERE {condition}]
[GROUP BY {attribute}]
    [HAVING {condition}]
[ORDER BY {attribute}]
```

where {argument} denotes you need to have at least one, and
where [argument] denotes a part that is optional and can be omitted

Domain Description Language (DDL)

creating databases

so we know how to use databases …

… but how do we create a database of our own?

multi-step process:

- conceptual database modelling phase
  data requirements are expressed using an ER model
- logical database design phase
  normalisation is used to remove redundancies, often by splitting tables, to make modifications to the information easier
- database description language (DDL)
  SQL is used to define and create the database

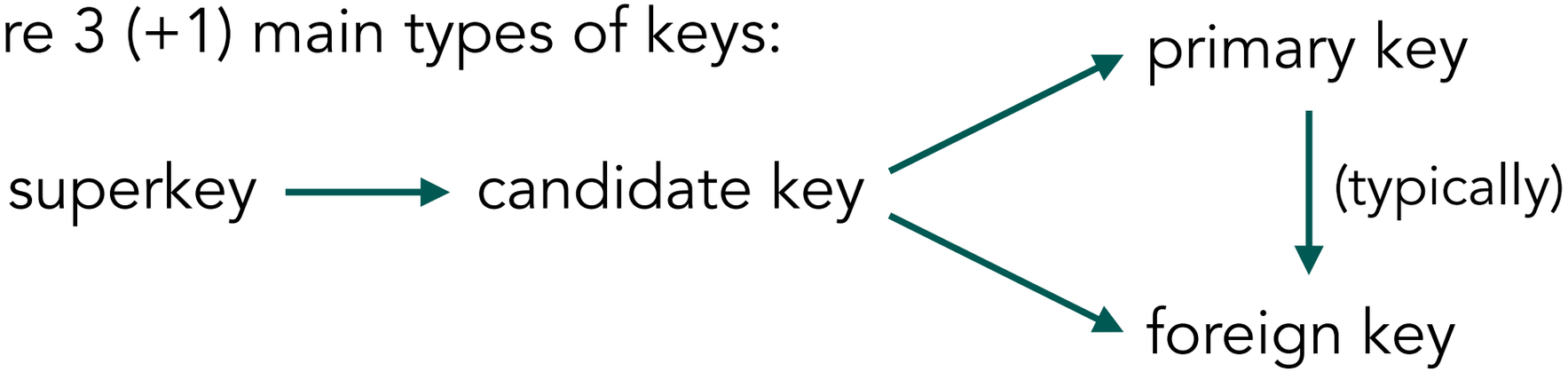
Domain Description Language (DDL)

(recap) keys: usage and types

keys allow to *uniquely identify tuples*

- **`key' component of any database!**
- can consist of one or more attributes
  simple key uses one attribute,
  composite key uses more than one
- help to express connection between relations

there are 3 (+1) main types of keys:



Domain Description Language (DDL)

(recap) keys: usage and types

meet: the *superkey*

some (or all) of the attributes of a relation schema that are *sufficient* to uniquely identify any given tuple

→ may include extraneous attributes

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000

Domain Description Language (DDL)

(recap) keys: usage and types

meet: the *superkey*

some (or all) of the attributes of a relation schema that are *sufficient* to uniquely identify any given tuple

yes *maybe* *maybe*

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000

yes

Domain Description Language (DDL)

(recap) keys: usage and types

the humble *candidate key*:

a *superkey* that is *minimal*, *i.e.* no excess attributes

no *maybe*

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000

yes

Domain Description Language (DDL)

(recap) keys: usage and types

the all-important *primary key*:

a *candidate key* chosen as the *principal means*
of identifying tuples in a relation: typically listed as first attribute

ideally, a value that *never* (or **very** rarely) changes

no

id	**name**	**dept_name**	**salary**
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000

yes

Domain Description Language (DDL)

(recap) keys: usage and types

the social *foreign key*:

an attribute that corresponds to the primary key of another relation and which is used to link tuples together

instructor

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
1657	Kim	Comp. Sci.	[redacted]
15151	Mozart	Music	40000
22222	Einstein	Physics	95000

referring relation

teaches

instructor_ID	course_id	semester	year
1657	CSC7072	1	2015
1657	CSC1023	2	2016
…	…	…	…

yes

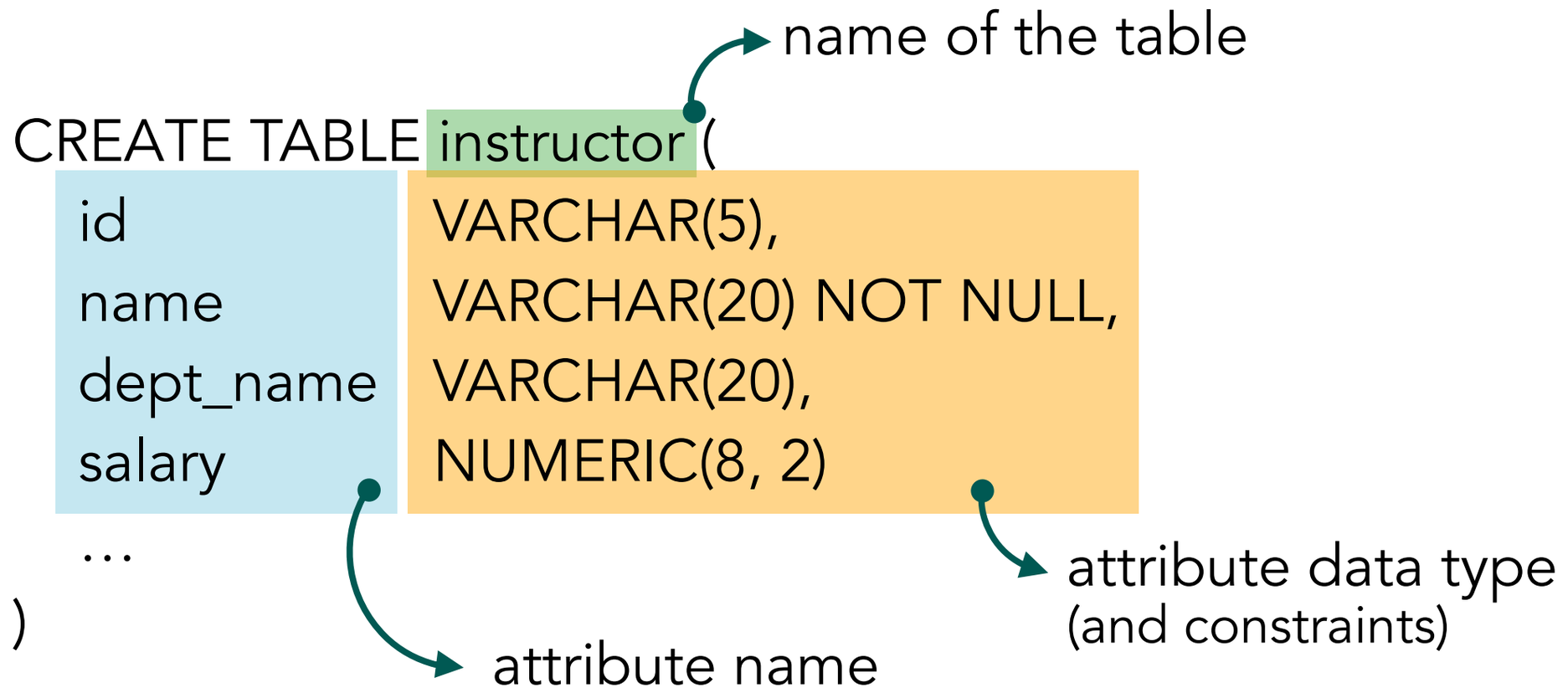
referenced relation

Domain Description Language (DDL)

adding tables using SQL

adding tables using SQL

special syntax for name and attributes:



Domain Description Language (DDL)

data types

general data types in SQL

- char(n) — fixed-length string with length n
- varchar(n) — variable-length string with maximum length n
- numeric(p,d) — fixed point number, with user-specified precision of p digits (plus a sign) and d digits mantissa
- int/smallint — (small) integer (machine dependent)
- real/float(n) — floating point (machine dependent, with n precision)
- date/time — a date / time such as 2005-7-27 / 09:00:30.75
- timestap — combination of date and time; 2005-7-27 09:00:30.75

numeric(p,d) is *significantly* slower than int/smallint/real/float, but avoids *horrors* of using floating point operations

Domain Description Language (DDL)

user-defined data types

user-defined data types in SQL

you can define your own types if needed:

```
CREATE TYPE dollars AS NUMERIC(12,2) FINAL

CREATE TABLE instructor (
    id              VARCHAR(5),
    name            VARCHAR(20) NOT NULL,
    dept_name       VARCHAR(20),
    salary          dollars
    …
)
```

Domain Description Language (DDL)

integrity constraints

integrity constraints

used to ensure the integrity of the data, *e.g.* NOT NULL
ensures a rollback of any changes if the consistency is violated

six (6) common constraints are:

- PRIMARY KEY({attribute})
- FOREIGN KEY({attribute}) REFERENCES (table)
- NOT NULL (prevents use of NULL for that attribute)
- UNIQUE({attribute}) (set of attributes that are candidate key)
- DEFAULT value (if missing, set it to *value*)
- CHECK(P) with P a predicate of what to check

of which PRIMARY KEY automatically applies NOT NULL as well

Domain Description Language (DDL)

integrity constraints: example

example of constraints:

```
CREATE TABLE instructor (
    id              VARCHAR(5),
    name            VARCHAR(20) NOT NULL,
    dept_name       VARCHAR(20),
    salary          dollars CHECK(salary > 29000),
    PRIMARY KEY (id),
    FOREIGN KEY (dept_name)
        REFERENCES department (dept_name)
        ON DELETE SET NULL
)
```

refers to *dept_name* in *department*
of course, needs to already exists

Domain Description Language (DDL)

referential constraints

referential constraints are important!

ensure that a value appearing in one relation for a given set of attributes also appears for the related attributes in the other relation

i.e. enforces that foreign keys are pointing to something that actually exists

```
CREATE TABLE instructor (
    …
    FOREIGN KEY (dept_name)
        REFERENCES department (dept_name)
        ON DELETE SET NULL
)
```

will not allow you to add an instructor unless the specified *dept_name* exists in *department*!

Domain Description Language (DDL)

cascading actions for referential constraints

cascading actions in referential constraints

specifies behaviour if original FK would be altered/deleted

two types:

- ON DELETE
- ON UPDATE

four options:

▪ CASCADE	also delete the tuple in the referenced relation
▪ SET DEFAULT	set value to DEFAULT on delete/update
▪ SET NULL	set value to NULL on delete/update
▪ (nothing)	prevent FK from being deleted/updated

Domain Description Language (DDL)

cascading actions: example

```
CREATE TABLE advisor (
    student_id        VARCHAR(5),
    instructor_id     VARCHAR(5),
PRIMARY KEY (student_id),
FOREIGN KEY (instructor_id)
    REFERENCES instructor (id)
    ON DELETE SET NULL,
FOREIGN KEY (student_id)
    REFERENCES student (id)
    ON DELETE CASCADE  )
```

Domain Description Language (DDL)

cascading actions: example cont.

instructor

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000

student

id	name	dept_name	tot_cred
12345	Shankar	Comp. Sci.	32
98765	Bourikas	Elec. Eng.	98

INSERT INTO advisor VALUES(34054, 10101)

fails; 34054 does not exist in *student*

INSERT INTO advisor VALUES(12345, 10101) **succeeds**

Domain Description Language (DDL)

cascading actions: example cont.

instructor

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000

advises

student_id	instructor_id
12345	10101

student

id	name	dept_name	tot_cred
12345	Shankar	Comp. Sci.	32
98765	Bourikas	Elec. Eng.	98

Domain Description Language (DDL)

cascading actions: example cont.

instructor

id	name	dept_name	salary
~~10101~~	~~Srinivasan~~	~~Comp. Sci.~~	~~65000~~
12121	Wu	Finance	90000

FOREIGN KEY (instructor_id)
REFERENCES instructor (id)
ON DELETE SET NULL

advises

student_id	instructor_id
12345	*null*

student

id	name	dept_name	tot_cred
12345	Shankar	Comp. Sci.	32
98765	Bourikas	Elec. Eng.	98

Domain Description Language (DDL)

cascading actions: example cont.

instructor

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000

FOREIGN KEY (student_id)
REFERENCES student (id)
ON DELETE CASCADE

advises

student_id	instructor_id
~~12345~~	~~10101~~

student

id	name	dept_name	tot_cred
~~12345~~	~~Shankar~~	~~Comp. Sci.~~	~~32~~
98765	Bourikas	Elec. Eng.	98

Domain Description Language (DDL)

domains vs types

user-defined data types in SQL

you can define your own domains if needed:

```
CREATE DOMAIN name AS VARCHAR(20) NOT NULL
```

not as widely supported as TYPE

DOMAIN allows you to also impose constraints:

```
CREATE DOMAIN degree_level varchar(10)
    CONSTRAINT degree_level_test
        CHECK (value in ('Bachelors', 'Masters', 'Doctorate') )
```

Domain Description Language (DDL)

integrity constraints

so … should I use constraints?

advisory: use them sparingly, wisely
well-used, great help; misused, a huge hindrance

always use PRIMARY KEY, FOREIGN KEY, NOT NULL, cascade actions

do use DEFAULT if there is a reasonable default value in the domain

be wary of CHECK: often far, *far* easier on programming level

do use VARCHAR, NUMERIC, DATE, TIME as needed

be wary of user-defined types, machine dependent types

Domain Description Language (DDL)

dropping/altering tables

one more thing …

you can also delete or change a table!

DROP TABLE instructor
drop the entire table *instructor*

ALTER TABLE instructor ADD gender VARCHAR(1)
add the attribute column *gender* using the specified type/domain

ALTER TABLE instructor DROP salary
drop the entire attribute column *salary*