MTAT.03.105
Introduction to Databases

Lecture #3

Data Types, Default values, Constraints

Ljubov Jaanuska
(ljubov.jaanuska@ut.ee)
Lecture 1. Summary

- SQL is a standard language for querying and manipulating data
- SQL comprises several functionalities: DDL, DML, DCL, TCL, DRL
- It works because it is optimized well!
- Number of standards
  - original ANSI SQL
  - updated in 1992 to SQL-92 or SQL2
  - most recent is SQL:1999 (also called SQL3)
  - extensions to SQL:1999, called SQL:2003
- DBMS vendors
  - all support ANSI SQL and largely SQL-92
  - some of SQL:1999 and SQL:2003
  - with variations and own extensions
Lecture 1. Summary

- **SQL commands** are not case sensitive:
  - Same: SELECT, Select, select

- **Values** can be case sensitive (depends on settings)
  - Different: ‘Seattle’, ‘seattle’

- Use single quotes for constants:
  - ‘abc’ - yes
  - “abc” - no
Lecture 1. Summary

- A **relation** or **table** is a multiset of tuples having the attributes specified by the schema.

- A **tuple** or **row** is a single entry in the table having the attributes specified by the schema (sometimes referred as a **record**).

- An **attribute** (or **column**) is a typed data entry present in each tuple in the relation.
Lecture 1. Summary

- **A relation schema** of a table is the table name, its attributes, and their types:

  ```
  Product(Pname: string, Price: float, Category: string, Manufacturer: string)
  ```

- **A key** is an attribute whose values are unique; we underline a key:

  ```
  Product(Pname: string, Price: float, Category: string, Manufacturer: string)
  ```

- **A database schema** is a collection of relation schemas, indexes, views, etc. (can be user dependent)
Lecture 1. Summary

SQL stands for **Structured Query Language**

<table>
<thead>
<tr>
<th>Product</th>
<th>PName</th>
<th>Price</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gizmo</td>
<td>$19.99</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td></td>
<td>Powergizmo</td>
<td>$29.99</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td></td>
<td>SingleTouch</td>
<td>$149.99</td>
<td>Canon</td>
</tr>
<tr>
<td></td>
<td>MultiTouch</td>
<td>$203.99</td>
<td>Hitachi</td>
</tr>
</tbody>
</table>

The number of tuples is the **cardinality** of the relation.

The number of attributes is the **arity** of the relation.
Lecture 1. CREATE TABLE

- CREATE TABLE <name> (column definition);

- Column definition:
  - Column name
  - Data type
  - [default values]
  - [constraints]

```sql
CREATE TABLE Person(
    name varchar(30) not null,
    surname varchar(100) not null,
    sex char(1) not null check (sex in ('m', 'f')),
    birthday date not null,
    entered datetime not null default current_timestamp,
    personal_id varchar(11),
    CONSTRAINT pk_person PRIMARY KEY (personal_id)
);
```
Lecture 2. What will you learn

- Data types
- Default values
- Constraints
- Create tables
- PCA development
Lecture 1. CREATE TABLE

• CREATE TABLE <name> (column definition);
  • Column definition:
    • Column name
    • Data type
    • [default values]
    • [constraints]
Data types: strings

**CHAR(n)** – fixed-length strings
- e.g. social security numbers or postal codes;
- max input data 32767 bytes

**VARCHAR(n)** – variable-length strings
- e.g. names and data that is likely to vary greatly in length
- max input data 32767 bytes

**LONG VARCHAR** – variable-length strings
- e.g. text, comments
- max input size 2 GB-1, in bytes \((2^{31} - 1)\)
Data types: integers

**TINYINT** – integers that require 1 byte of storage
  - 0 to 255

**SMALLINT** – integers that require 2 bytes of storage
  - -32768 to 32767

**INTEGER** – integers that require 4 bytes of storage
  - -2147483648 to 2147483647

**BIGINT** – integers requiring 8 bytes of storage
  - -9223372036854775808 to 9223372036854775807
Data types: decimal numbers

**DECIMAL** (number of digits in the expression, number of digits after the decimal point)
- DECIMAL(5,3) e.g. 12.345; 34.500;
- Max storage size varies based on the precision (5, 9, 13, 17 bytes)

**NUMERIC** (precision, scale)

**FLOAT** (number of digits after the decimal point)
- FLOAT(3) e.g. 12.345; 34.500;
- 4 bytes if precision is not provided, but can be extended to 8 bytes
Data types: decimal numbers

**REAL** – floating-point numbers stored in 4 bytes
  • range between $-3.402823 \times 10^{38}$ to $3.402823 \times 10^{38}$, with numbers close to zero as small as $1.175494351 \times 10^{-38}$
  • accurate to 7 significant digits

**DOUBLE** – floating-point numbers stored in 8 bytes
  • range between $-1.79769313486231 \times 10^{308}$ to $1.79769313486231 \times 10^{308}$, with numbers close to zero as small as $2.22507385850721 \times 10^{-308}$
  • accurate to 15 significant digits
Data types: binary

**BIT** - values 0 or 1

**BINARY** - stores binary data of a specified maximum length (max length 32767 bytes)

**LONG BINARY** - stores binary data of arbitrary length (max value 2GB -1)
CREATE TABLE Club

- Create table Club with two columns: club_id and name.
- In column club_id, the data type must be to contain club id numbers.
- In column name, set the data type as varchar and the max length of names is 100 symbols.

Hints:
Club
  club_id INTEGER NOT NULL DEFAULT AUTOINCREMENT
  c_name VARCHAR(100) NOT NULL
Lecture 1. CREATE TABLE

- CREATE TABLE <name> (column definition);
  - Column definition:
    - Column name
    - Data type
    - [default values]
    - [constraints]
DEFAULT values

AUTOINCREMENT - the feature assigns each new row a unique value larger than any other value in the column

• e.g. to record purchase order numbers, to identify customer service calls or other entries where an identifying number is required

• club_id INTEGER NOT NULL DEFAULT AUTOINCREMENT
DEFAULT values

**TIMESTAMP** - default value is used to record the local date and time of day when a row in a table was last *modified*

**CURRENT_TIMESTAMP** - combines CURRENT DATE and CURRENT TIME to form a TIMESTAMP value containing the year, month, day, hour, minute, second and fraction of a second

- The main difference:
  - DEFAULT_TIMESTAMP columns are set at both INSERT and UPDATE
  - DEFAULT_CURRENT_TIMESTAMP columns are set only at INSERT
CREATE TABLE Player

Create table Player with 5 columns:

Player
  id integer not null default autoincrement
  p_name varchar (50) not null
  surname varchar(50) not null
  player_id varchar(11)
  club_id integer
Lecture 1.  CREATE TABLE

- CREATE TABLE <name> (column definition);
  - Column definition:
    - Column name
    - Data type
    - [default values]
    - [constraints]
Constraints

• SQL **constraints** are used to specify **rules for the data** in a table.

• If there is any violation between the constraint and the data action, the action is aborted by the constraint.

• Constraint types:
  • UNIQUE
  • PRIMARY KEY
  • FOREIGN KEY
  • NOT NULL
  • CHECK
Constraints

UNIQUE - constraint **uniquely** identifies each record in a database table

PRIMARY KEY - **uniquely** identifies each record in a database table
  - primary key column cannot contain NULL values
  - automatically has a UNIQUE constraint defined on it

Difference between UNIQUE and PRIMARY KEY:
  - Note that one can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table

FOREIGN KEY in one table points to a PRIMARY KEY in another table.

NB: Primary and foreign keys will be considered in detail on session 6
Constraints

**NOT NULL** constraint enforces a column to NOT accept NULL values

**CHECK** is used to limit the value range that can be placed in a column
- sex **CHAR(1) NOT NULL CHECK(sex IN('M','F'))**
Constraints

Constraints can be specified:

• inside the CREATE TABLE statement:

```sql
CREATE TABLE Person(
    name varchar(30) not null,
    surname varchar(100) not null,
    sex char(1) not null check (sex in ('m', 'f')),
    birthday date not null,
    entered datetime not null default current_timestamp,
    personal_id varchar(11),
    CONSTRAINT pk_person PRIMARY KEY (personal_id)
);
```

• after the table is created inside the ALTER TABLE statement:

```sql
ALTER TABLE Table_name ADD CONSTRAINT Constraint_name Constraint_TYPE (column_name)
```

```sql
-> ALTER TABLE Player ADD CONSTRAINT pk_player PRIMARY KEY (id);
-> ALTER TABLE Club ADD CONSTRAINT un_name UNIQUE (c_name);
```
Constraints

Constraints can be deleted

- by names
  \[
  \text{ALTER TABLE} \ Club \ \text{DROP CONSTRAINT} \ \text{un\_name};
  \]

- by constraint definition
  \[
  \text{ALTER TABLE} \ Club \ \text{DROP UNIQUE} \ (c\_name);
  \]
Add columns into table

ALTER TABLE <Table_name> ADD {column_definition};

-> Add column into table Club:

ALTER TABLE Club ADD city VARCHAR (50) NOT NULL DEFAULT 'Tartu'
Import data from a file

Download and unzip files from Moodle or courses.cs.ut.ee

NB! Pay attention to the columns in the .txt file and to the structure of the table

Game

INPUT INTO Club FROM 'C:/club.txt' FORMAT ASCII DELIMITED BY '\x09' (club_id, c_name);

INPUT INTO Player FROM 'C:/player.txt' FORMAT ASCII DELIMITED BY '\x09' (id, p_name, surname, club_id);
At home:
CREATE TABLE Tournament

Tournament
  tournament_id integer not null default autoincrement
  name varchar(100) not null unique
  venue varchar(100)
  start_date date not null
  end_date date
CREATE TABLE Game

Game
game_id integer not null default autoincrement primary key
tournament_id integer not null
begins datetime not null default current timestamp
ends datetime
white integer not null
black integer not null
white_result smallint
black_result smallint
summary varchar(5000)
Add the following constraints:
• Table *Player*: ID primary key
• Table *Player*: unique combinations of names and surnames
• Table *Club*: ID primary key
• Table *Club*: unique names
• Table *Game*: game results for both players (white and black chess pieces) must be 0, 1 or 2
• Table *Game*: sum of both player results must be equal to 2
At home:
Import data

• Import data into tables *Tournament*
• Import data into tables *Game*
  • NB! Pay attention to the columns in the .txt file and to the structure of the table *Game*

Present PCA.db and PCA.log files via Moodle by 28th of February.
Tables

- Player
- Club
- Game
- Tournament