

B0B36DBS, BD6B36DBS: **Database Systems**

<http://www.ksi.mff.cuni.cz/~svoboda/courses/182-B0B36DBS/>

Lecture 3

SQL: Data Definition

Martin Svoboda

martin.svoboda@fel.cvut.cz

5. 3. 2019

Czech Technical University in Prague, Faculty of Electrical Engineering

Outline

- **SQL**
 - **Data definition**
 - Definition of tables
 - Data types
 - Integrity constraints
 - Schema modification
 - **Data manipulation**
 - Insertion
 - Updates
 - Deletion

Structured Query Language (SQL)

Structured Query Language

- **SQL**

- Standard language for accessing relational databases
 - **Data definition (DDL)**
 - Creation of table schemas and integrity constraints
 - **Data manipulation (DML)**
 - Querying
 - Data insertion, deletion, updates
 - Transaction management
 - Modules (programming language)
 - Database administration

Structured Query Language

- **SQL standards**

- Backwards compatible
- ANSI/ISO
 - **SQL-86** – intersection of IBM SQL implementations
 - SQL-89 – small revision, integrity constraints
 - **SQL-92** – schema modification, transactions, set operators, new data types, cursors, referential integrity actions, ...
 - **SQL:1999** – recursive queries, triggers, object-relational features, regular expressions, types for full-text, images, spatial data, ...
 - **SQL:2003** – SQL/XML, sequence generators
 - SQL:2006 – other extensions of XML, integration of XQuery
 - SQL:2008
 - SQL:2011 – temporal databases

Structured Query Language

- **Commercial systems**

- Current implementations at different standard levels
 - Most often SQL:1999, SQL:2003
- However (and unfortunately)...
 - **Some extra proprietary features supported**
 - **Some standard features not supported**
 - Even syntax may differ
 - And so data migration is usually not straightforward
- Specific extensions
 - Procedural, transactional and other functionality, e.g.,
TRANSACTION-SQL (Microsoft SQL Server), PL/SQL (Oracle)

SQL Syntax Diagrams

- **Syntax (railroad) diagrams**

- Graphical representation of context-free grammars
 - I.e. a practical approach how to describe languages (such as SQL) in a graphical and user-friendly way
- Technically...
 - Directed graph representing an automaton accepting SQL
 - Terms in diagrams:
 - Capital letters on blue – keywords
 - Small letters on green – literals
 - Small letters on orange – subexpressions



SQL: Schema Definition

Table Creation

- **CREATE TABLE**

- Construction of a table schema (and an empty table)
 - **Table name**
 - Definition of **table columns**
 - Together with their column-scope integrity constraints
 - Definition of **table-scope integrity constraints**

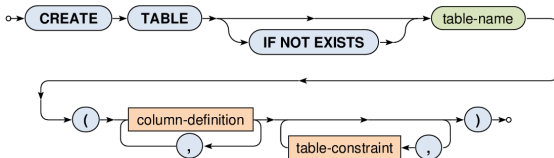


Table Creation

- **CREATE TABLE**

- Definition of table columns

- **Column name**

- **Data type**

- **Default value**

- When a new row is about to be inserted and not all its values are specified, then the default values are used (if defined)

- Definition of column-scope IC

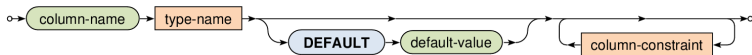


Table Creation

- **Example**

- Simple table without integrity constraints

```
CREATE TABLE Product (  
    Id INTEGER,  
    Name VARCHAR(128) ,  
    Price DECIMAL(6,2) ,  
    Produced DATE,  
    Available BOOLEAN DEFAULT TRUE,  
    Weight FLOAT  
);
```

Data Types

- Available data types
 - Precise numeric types
 - **INTEGER**, INT, SMALLINT, BIGINT
 - **DECIMAL**(precision, scale)
 - Precision = number of all digits (including decimal digits)
 - Scale = number of decimal digits
 - Approximate numeric types
 - FLOAT, REAL, DOUBLE PRECISION – real numbers
 - Logical values
 - **BOOLEAN**

Data Types

- Available data types
 - Character strings
 - **CHAR**(length), **CHARACTER**(length) – fixed-length strings
 - Shorter strings are automatically right-padded with spaces
 - **VARCHAR**(length), **CHARACTER VARYING**(length)
 - Strings of a variable length
 - Temporal types
 - **DATE**, **TIME**, **TIMESTAMP**
- **Type conversions**
 - Meaningful conversions are defined automatically
 - Otherwise see **CAST**...

Data Types

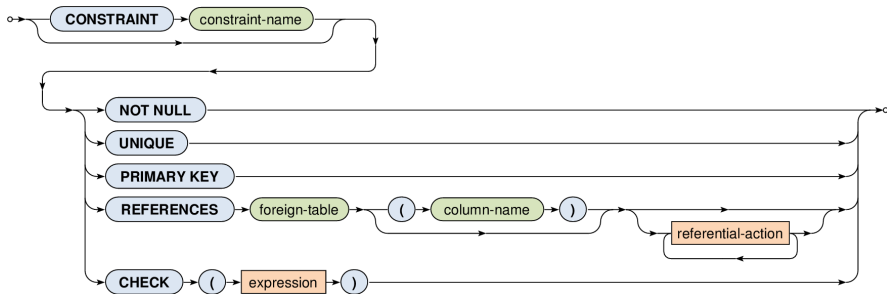
- **Example**

- Simple table without integrity constraints

```
CREATE TABLE Product (  
    Id INTEGER,  
    Name VARCHAR(128),  
    Price DECIMAL(6,2),  
    Produced DATE,  
    Available BOOLEAN DEFAULT TRUE,  
    Weight FLOAT  
);
```

Integrity Constraints

- **Column integrity constraints**
 - Allow us to limit domains of the allowed values



Integrity Constraints

- Column integrity constraints
 - **NOT NULL**
 - Values must not be `NULL`
 - **UNIQUE**
 - All values must be distinct
 - But can there be just one or multiple `NULL` values?
 - **PRIMARY KEY**
 - Only one primary key is allowed in a table!
 - Equivalent to NOT NULL + UNIQUE

Integrity Constraints

- Column integrity constraints
 - **FOREIGN KEY**
 - Referential integrity
 - Values from the referencing table must also exist in the referenced table
 - `NULL` values are ignored
 - Only unique / primary keys can be referenced
 - **CHECK**
 - Generic condition that must be satisfied
 - However, only values within a given row may be tested

Integrity Constraints: Example

```
CREATE TABLE Producer (  
    Id INTEGER PRIMARY KEY,  
    Name VARCHAR(128),  
    Country VARCHAR(64)  
);
```

```
CREATE TABLE Product (  
    Id INTEGER CONSTRAINT IC_Product_PK PRIMARY KEY,  
    Name VARCHAR(128) UNIQUE,  
    Price DECIMAL(6,2) CONSTRAINT IC_Product_Price NOT NULL,  
    Produced DATE CHECK (Produced >= '2015-01-01'),  
    Available BOOLEAN DEFAULT TRUE NOT NULL,  
    Weight FLOAT,  
    Producer INTEGER REFERENCES Producer (Id)  
);
```

Integrity Constraints: Example

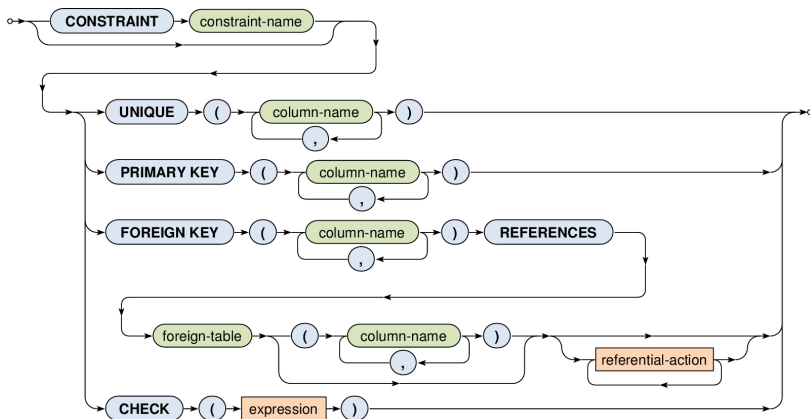
- **Example**

- Referential integrity within a single table

```
CREATE TABLE Employee (  
    Id INTEGER PRIMARY KEY,  
    Name VARCHAR(128),  
    Boss INTEGER REFERENCES Employee (Id)  
);
```

Integrity Constraints

- Table integrity constraints



Integrity Constraints

- **Table integrity constraints**

- Analogous to column IC, just for multiple columns, i.e. for tuples of values

- **UNIQUE**

- **PRIMARY KEY**

- **FOREIGN KEY**

- Tuples containing at least one `NULL` value are ignored

- **CHECK**

- Even with more complex conditions testing the entire tables
 - However, table integrity constraints are considered to be satisfied on empty tables (by definition, without evaluation)
 - See `CREATE ASSERTION...`

Integrity Constraints: Example

```
CREATE TABLE Producer (  
    Name VARCHAR(128),  
    Country VARCHAR(3),  
    CONSTRAINT IC_Producer_PK PRIMARY KEY (Name, Country)  
);
```

```
CREATE TABLE Product (  
    Id INTEGER PRIMARY KEY,  
    ...  
    ProducerName VARCHAR(128),  
    ProducerCountry VARCHAR(3),  
    CONSTRAINT IC_Product_Producer_FK  
        FOREIGN KEY (ProducerName, ProducerCountry)  
        REFERENCES Producer (Name, Country)  
);
```

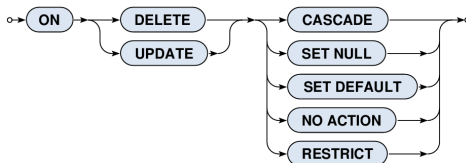
Referential Integrity

- **Referential actions**

- When an operation on the referenced table would cause violation of the foreign key in the referencing table...
 - I.e. value of the foreign key of at least one row in the referencing table would become invalid as a result
- ... then...
 - this operation is blocked and an error message is generated
 - but if a referential action is defined, it is triggered...

Referential Integrity

- **Referential actions**



- **Triggering situations**

- **ON UPDATE, ON DELETE**

- When the action is triggered
 - Once again, these are considered to be operations on the referenced table

Referential Integrity

- Referential actions
 - **CASCADE**
 - Row with the referencing value is updated / deleted as well
 - **SET NULL** – referencing value is set to `NULL`
 - **SET DEFAULT** – referencing value is set to its default
 - **NO ACTION** – default – no action takes place
 - I.e. as if no referential action would be defined at all
 - **RESTRICT** – no action takes place as well...
 - However, the integrity check is performed at the beginning, i.e. before the operation is even tried to be executed
 - ... and so triggers or the operation itself have no chance to remedy the situation even if they could be able to achieve such a state (and so **RESTRICT** is different to **NO ACTION**)

Referential Integrity: Example

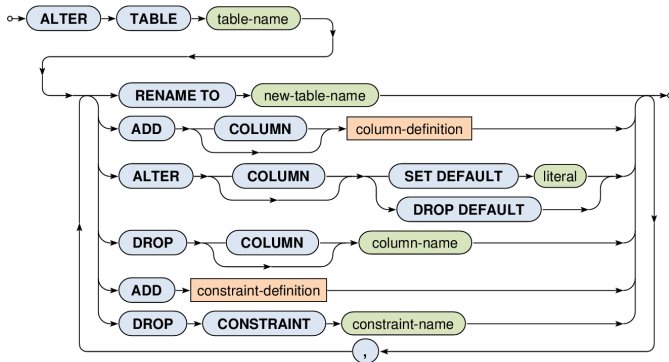
```
CREATE TABLE Producer (  
    Id INTEGER PRIMARY KEY,  
    Name VARCHAR(128),  
    Country VARCHAR(64)  
);
```

```
CREATE TABLE Product (  
    Id INTEGER PRIMARY KEY,  
    ...  
    Producer INTEGER  
        REFERENCES Producer (Id) ON DELETE CASCADE  
);
```

Schema Modification

- **ALTER TABLE**

- Addition/change/removal of table columns/IC



Schema Modification

- **DROP TABLE**

- Complementary to the table creation
 - I.e. table definition as well as table content are deleted



SQL: Data Manipulation

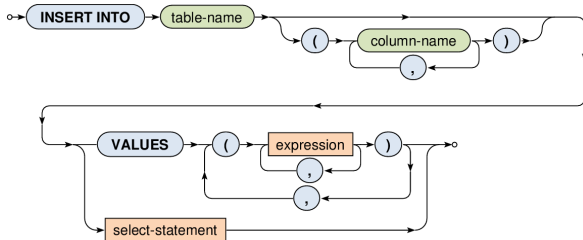
SQL Data Manipulation

- Data manipulation language
 - Data modification
 - **INSERT INTO** – insertion of rows
 - **DELETE FROM** – deletion of rows
 - **UPDATE** – modification of rows
 - Data querying
 - **SELECT**... *the next lecture*

Data Insertion

- **INSERT INTO**

- Insertion of new rows into a table
 - ...by an explicit enumeration / from a result of a selection
 - Default values are assumed for the omitted columns



Data Insertion: Example

```
CREATE TABLE Product (  
    Id INTEGER PRIMARY KEY,  
    Name VARCHAR(128) UNIQUE,  
    Price DECIMAL(6,2) NOT NULL,  
    Produced DATE,  
    Available BOOLEAN DEFAULT TRUE,  
    Weight FLOAT,  
    Producer INTEGER  
);
```

INSERT INTO Product

```
VALUES (0, 'Chair1', 2000, '2015-05-06', TRUE, 3.5, 11);
```

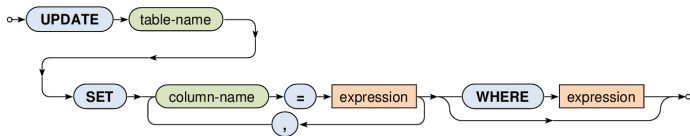
INSERT INTO Product

```
(Id, Name, Price, Produced, Weight, Producer)  
VALUES (1, 'Chair2', 1500, '2015-05-06', 4.5, 11);
```


Data Updates

- **UPDATE**

- Modification of existing rows in a table
 - Only rows matching the given condition are considered
- Newly assigned values can be...
 - `NULL`, literal, value given by an expression, result of a scalar subquery



Data Updates: Example

```
CREATE TABLE Product (  
    Id INTEGER PRIMARY KEY,  
    Name VARCHAR(128) UNIQUE,  
    Price DECIMAL(6,2) NOT NULL,  
    Produced DATE,  
    Available BOOLEAN DEFAULT TRUE,  
    Weight FLOAT,  
    Producer INTEGER  
);
```

UPDATE Product

```
SET Name = 'Notebook'  
WHERE (Name = 'Laptop');
```

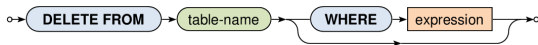
UPDATE Product

```
SET Price = Price * 0.9  
WHERE (Produced < '2015-01-01');
```

Data Deletion

- **DELETE FROM**

- Deletion of existing rows from a table
 - Only rows matching the given condition are considered



Data Deletion: Example

```
CREATE TABLE Product (  
    Id INTEGER PRIMARY KEY,  
    Name VARCHAR(128) UNIQUE,  
    Price DECIMAL(6,2) NOT NULL,  
    Produced DATE,  
    Available BOOLEAN DEFAULT TRUE,  
    Weight FLOAT,  
    Producer INTEGER  
);
```

```
DELETE FROM Product  
    WHERE (Price > 2000);
```

```
DELETE FROM Product;
```

