1 SQL

The **Structured Query Language** (SQL) is a **declarative** (goal explicit, algorithm implicit) query language. It was first standardized in 1986, and a major revision was done in 1992 (SQL-92 standard), which is largely the revision that is covered in the textbook.

2 Data Model

An SQL **table** is a **multiset** of tuples, meaning that duplicates are allowed.

2.1 Data Types

Tuples consist of values from specified domains

- **char(n)** is a fixed-length string
- **varchar(n)** is a variable-length string
- **int**, **real**, and **double** are the standard numeric types
- **numeric(p,d)** is a fixed-point numeric type with 'p' (precision) digits, where 'd' come after the decimal point, e.g. numeric(3,1) $\rightarrow$ xx.x .
- **date** and **time**

2.2 NULL Values

NULL values are interpreted as missing information, so:

- **arithmetics** with NULL yields NULL
- **comparison** with NULL yields UNKNOWN (neither TRUE nor FALSE)
- **logical connectives** behave as expected, FALSE AND UNKNOWN = FALSE, and TRUE OR UNKNOWN = TRUE, etc.
- **aggregates** ignore NULL values

3 Data Definition Language

Tables are created using the **CREATE TABLE** command, e.g.:

```
CREATE TABLE  person
  (name varchar(30) NOT NULL,
   age  integer,
   cpr  integer,
   PRIMARY KEY (cpr),
   CHECK age > 18)
```

CREATE TABLE defines the **schema** of the table, any **column default** values, **column constraints** such as:

NOT NULL must have value

UNIQUE no duplicate values of this attribute are allowed

PRIMARY KEY (NOT NULL and UNIQUE)

CHECK(predicate) is a generic column constraint, which is checked for each table update, similar to a class invariant ('CHECK' is ignored by mySQL).

Referential integrity ensures that references to other tables are consistent, using the FOREIGN KEY clause.

Tables can be altered using the **ALTER TABLE** command. It can be used to add, change, or remove attributes or constraints.

```
ALTER TABLE person ADD birthday TIME;
ALTER TABLE person CHANGE birthday birthday DATE;
ALTER TABLE person DROP birthday;
```

Tables can be removed from the database using the **DROP TABLE** command.

4 Data Manipulation Language

4.1 SELECT

SELECT is used for retrieving data from the database. The basic syntax is

```
SELECT attribute1, attribute2
FROM relation1, relation2
WHERE predicate.
```

In terms of relational algebra, this is actually a combination of a projection, a selection, and a cartesian product

$$\Pi_{\text{attribute1,attribute2}}(\sigma_{\text{predicate}}(\text{relation1} \times \text{relation2})).$$

I will explain the individual clauses in more detail.

The **SELECT** clause projects the listed attributes from the results of the selection, '*' denotes all attributes. SELECT returns a **multiset** (duplicates are allowed) containing the results. If duplicate tuples should not appear, we may use the **SELECT DISTINCT** form of the command. SELECT evaluates arithmetic expressions, e.g. 'SELECT age+2 FROM person' returns a table with one column named 'age+2' and the age values + 2. **Renaming** of attributes is performed using the **AS** clause, e.g. 'SELECT borrower.loan_number AS loan_id FROM borrower', thus creating a new table with attribute loan_id. the AS clause can also be used for defining **tuple variables**, as in

```
SELECT DISTINCT p1.firstname, p1.lastname
       FROM person AS p1, person AS p2
       WHERE p1.firstname = p2.firstname,
```

that finds the full names of persons with the same first name.

The **FROM** clause performs a cartesian product between the tables listed.

The **WHERE** clause specifies a logical predicate which must hold for all tuples in the result set. **Arithmetical expressions** are allowed in the predicate. Comparison is done with the usual arithmetic relations $<$, $<=$, $=$, $>=$, $>$, and $!=$. Equality checks on **strings** are also allowed. Predicates can be combined using the **logical connectives**, 'AND', 'OR', and 'NOT'.

Tuples can be ordered using the **ORDER BY** clause which has a list of attributes, on which to order the result set, and the **DESC** and **ASC** (default) keywords, specifying ascending or descending order.

Standard **set operations** can be performed on the result sets from SELECT operations with **UNION**, **INTERSECT**, and **EXCEPT**, e.g. (SELECT * FROM person) INTERSECT (SELECT * FROM customer). **Set membership** can be checked with 'SELECT ... WHERE name IN (SELECT name FROM ...)' or 'SELECT ... name IN ('Smith', 'Jones').

The first order logic operators $\exists$ and $\forall$ can also be used in SQL; **Existence** of at least one tuple that makes a predicate true, can be checked, e.g. with 'SELECT ... WHERE value $>$ SOME (SELECT value FROM ...) (the tuples where value are not the smallest), and correspondingly, you can check **all tuples** for a predicate with 'SELECT ... WHERE value $>$ ALL (SELECT value FROM ...)' (larger than all values in the set).

An **aggregate function** is a function that compute a single result value from a set of input values. SQL includes 5 common aggregate functions:

- AVG (average)
- COUNT (count)
- MAX
- MIN
- SUM

As an example,

```
SELECT city, AVG(salary) FROM person GROUP BY city,
```

returns for each city the average salary of persons from that city. The results can be filtered using the **HAVING** clause, which is applied after the groups are formed, and thus allows aggregates, e.g.

```
SELECT city, AVG(salary) FROM person
       GROUP BY city HAVING AVG(salary) > 100000.
```

4.2 Data Modification

INSERT Examples:

```
INSERT INTO person VALUES ('Jakob Schmid',30,290276)
INSERT INTO person
       SELECT name,age,cpr FROM customer WHERE name="jakob"
```

The **DELETE** command has a similar syntax to SELECT: 'DELETE FROM table WHERE predicate'. The **UPDATE** command can be used like 'UPDATE table SET attr=value WHERE ...'.

4.3 Views

Views are defined in SQL using the **CREATE VIEW** command, e.g. 'CREATE VIEW elite_customer AS SELECT name,age FROM customer WHERE status="elite"'. A view can be used as a table for all purposes. Updating the view also updates the table from which the view is created.