Today’s lecture

Based on Chapter 3 and 7 in KBL. All of Chapter 3 and 7 is included in the curriculum.

- Basic concepts and terminology
- Integrity constraints
  - Key constraints
  - Referential integrity
  - Semantic constraints
- SQL Data Definition Language (DDL)

Physical, conceptual and external schemas

Physical schema: How data is stored physically, including indices
Conceptual schema: Which data is stored and how it is structured (1 per db)
External schema: Customized schema for (group of) users, describing data and structure (many for the same db)

The DBMS maps data between conceptual and physical level automatically. I.e. if the physical storage is changed, the application can still be unchanged.

Conceptual data independence: When applications work with external schemas, i.e. independent of both physical and conceptual schemas.

Data model

1. Conceptual and external schemas
   - Described by DDL, data definition language
2. Constraints
   - Conditions that data must satisfy
   - Described by DDL
3. Operations on data
   - Described by DML, data manipulation language
SDL, storage definition language used to influence the physical schema
SQL: DDL, DML, and SDL

The relational model

- Introduced by E. F. Codd in 1970
- Based on mathematical logic
- Queries can be analyzed and transformed to equivalent expressions
  - Nice property for a query optimizer
- Most commercial DBMSs are based on the relational model
Terminology

**Relation instance**: a table with a set of rows and a set of named columns

- **Row**: Tuple
- **Named column**: Attribute
- **Number of tuples**: Cardinality of relation
- **Number of columns**: Arity of the relation
- There is no order of tuples and attributes

Terminology cont.

Attributes have values from an **attribute domain** (e.g. int, string,...)

- Values are atomic, i.e. can not look at internal structure of values (e.g. 3rd character in a string), in theory
- **In practice**: can do some things or everything (object relational)

Relation schema

Consists of:

1. Name of relation
2. Attributes
   - Name
   - Domain name (type)
     • Type constraint
3. Integrity constraints

SQL: CREATE TABLE

```
CREATE TABLE CAR (
    Regnr VARCHAR(8),
    Ownerid INTEGER,
    Color VARCHAR(15))
```

SQL: CREATE DOMAIN

```
CREATE DOMAIN ColorType CHAR(10)
    CHECK ( VALUES IN ('blue', 'white', 'pink', 'lilac', 'green') )
```

Any SQL statement producing a unary table can be used as expression.

Not supported by MySQL.

Integrity constraints (IC)

Restricts the relational instances of a schema

An instance of a database is legal if all ICs are satisfied

Types of constraints:

- Key constraints
- Referential integrity
- Semantic constraints

The DBMS is responsible for making sure the ICs are not violated.

Not all DBMSs support all types of ICs.
Key constraints, definition

A key constraint $\text{key}(K)$ associated with a relational schema $S$, consists of a subset, $K$ (called a key), of attributes in $S$ satisfying:

Uniqueness property:
Let $s$ be an instance of $S$. $s$ does not contain a pair of distinct tuple whose values agree on all the attributes in $K$.

Problem session

What can be said about the (candidate) keys of the relation shown below?

<table>
<thead>
<tr>
<th>saloID</th>
<th>salomark</th>
<th>regno</th>
<th>make</th>
<th>office</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>B. H.</td>
<td>VY 34718</td>
<td>Opel</td>
<td>City</td>
</tr>
<tr>
<td>53</td>
<td>W. G.</td>
<td>PQ 11112</td>
<td>Ford</td>
<td>Redwood</td>
</tr>
<tr>
<td>87</td>
<td>B. H.</td>
<td>MX 36791</td>
<td>Ford</td>
<td>City</td>
</tr>
<tr>
<td>99</td>
<td>L. B. H.</td>
<td>AB 12345</td>
<td>Porche</td>
<td>City</td>
</tr>
</tbody>
</table>

SQL: Primary key

CREATE TABLE CAR (  
  Regnr VARCHAR(8) NOT NULL,  
  Ownerid INTEGER,  
  Color VARCHAR(15),  
  PRIMARY KEY (Regnr),  
  UNIQUE (Ownerid, Color) )

Here we assume that one person can only have one car of each color.

Referential integrity

Definition, referential integrity:  
When a tuple has a reference to another tuple, then the referenced tuple must exist.

STUDENT(Id:INT, Name:STRING)  
Key: {Id}

TRANSCRIPT(StudId:INT, CrsCode:STRING, Grade:STRING)  
Key: {StudId, CrsCode}

StudId in TRANSCRIPT references Id in STUDENT

Often the referenced value is the primary key. Efficient to check, as it is indexed.

Foreign key

An attribute $A$ is a foreign key of $R_1$ referring to attribute $B$ in $R_2$, if whenever there is a non-NULL value $v$ of $A$, there is a tuple of $R_2$ in which $B$ has value $v$, and $B$ is a key of $R_2$.

The foreign key can be composite.
Foreign key constraint

We can tell the DBMS that StudID is a foreign key using the syntax:

```sql
FOREIGN KEY StudID references STUDENT(id)
```

What happens when a referenced tuple in STUDENT is changed or deleted?

SQL: Foreign key trigger

(MySQL syntax:)

```sql
CONSTRAINT `Oid` FOREIGN KEY (`Ownerid`)
REFERENCES `Owner` (`Id`) ON DELETE
SET NULL ON UPDATE CASCADE)
``` 

ON DELETE/UPDATE

SET NULL: Set reference to NULL
NO ACTION: Update or delete rejected
CASCADE: Delete/update the reference

Problem session

Consider the following relations:

- `PERSON(Cpr,Name,Birthday)`
- `ADDRESS (Id,Street,Number,Zip,City)`
- `LIVESAT(Cpr,AddressId)`
- `PHONE(SubCpr,Number,Type,AddressId)`

What are suitable primary/foreign keys?
Are there other candidate keys?
What should happen when an address is deleted?

Semantic constraints

Constraints dealing with the meaning of data (instead of structure).

Derived from the application area.

Examples:

- Number of students registered for a course can't exceed the number of seats in the classroom for that course.
- End date must be after start date.

SQL: CHECK

```sql
CREATE TABLE CAR (
 Regnr VARCHAR(8) NOT NULL,
 Ownerid INTEGER,
 Color VARCHAR(15),
 PRIMARY KEY (Regnr),
 CHECK (Ownerid>999 AND NOT Color='Lilac' ) )
```

Two semantic constraints that say that Ownerids always have at least 4 digits and that cars can not be lilac.

Note: CHECK not implemented in MySQL.
SQL: ASSERTION CHECK

The CHECK condition in a CREATE TABLE statement is checked every time a tuple is changed, and is intrarelational.

**CHECK conditions can involve other relations as well, interrelational.**

```sql
CREATE ASSERTION EnoughCars
CHECK (2<(SELECT COUNT(*) FROM Cars)/(SELECT COUNT(*) FROM Owners))
```

```sql
CREATE ASSERTION ColorfulCars
CHECK ((SELECT DISTINCT Color FROM Cars)> (SELECT COUNT(*) FROM Owners))
```

Reactive constraints, triggers

Reactive constraint: What to do if a certain event occurs

```sql
CREATE TRIGGER AddStudToAStudents
AFTER UPDATE OF Grade ON Transcript
FOR EACH ROW
WHEN Grade='A'
INSERT INTO AStudents VALUES Studid,CrsCode,NOW()
```

Triggers, general form

- Before or after event
- Event is insert, delete, or update
- Can specify which columns to react to
- Can refer to old and new values, e.g. can check how a value is changed or reuse an old value instead of the new.
- Can specify to do the check each time a row is changed or after each statement.
- Can specify a precondition.
- Action to be executed can be written in SQL combined with other statements like if-then -else and loops (SQL/PSM)

Quotes from MySQL 5.0 Manual

"The CHECK clause is parsed but ignored by all storage engines."

"Currently, triggers are not activated by cascaded foreign key actions. This limitation will be lifted as soon as possible."

Things not covered in the lecture

- Syntax for inserting, deleting, and updating table definitions, assertions, and domains
- Views – Virtual tables, similar to a schema, but uses data in other schemas. "Created" when used.
- Access control – how to give specific users access to only a part of the database
- Triggers – much more in Chapter 7