

Implement a Relational Model

CS5200 DBMS
Bruce Chhay

Implement a Relational Model

L1: Implement a Relational Database

CREATE TABLE

```
CREATE TABLE tbl_name (  
    col_name data_type [col_opts],  
    col_name ...,  
    CONSTRAINT constraint_name [constraint_options],  
    CONSTRAINT ...  
);
```

CREATE TABLE

```
CREATE TABLE tbl_name (  
  col_name data_type [col_opts],  
  col_name ...,  
  CONSTRAINT constraint_name [constraint_options],  
  CONSTRAINT ...  
);
```

CREATE TABLE

```
CREATE TABLE tbl_name (  
    col_name data_type [col_opts],  
    col_name ...,  
    CONSTRAINT constraint_name [constraint_options],  
    CONSTRAINT ...  
);
```

Enforcing Data Integrity

- Entity integrity: Data types, column options, and primary key constraints
- Referential integrity: Foreign key constraints
- Business rule integrity: Triggers, application-specific

```
CREATE TABLE tbl_name (  
    col_name data_type [col_opts],  
    col_name ...,  
    CONSTRAINT constraint_name [constraint_options],  
    CONSTRAINT ...  
);
```

Enforcing Data Integrity

- Entity integrity: Data types, column options, and primary key constraints
- Referential integrity: Foreign key constraints
- Business rule integrity: Triggers, application-specific

```
CREATE TABLE tbl_name (  
    col_name data_type [col_opts],  
    col_name ...,  
    CONSTRAINT constraint_name [constraint_options],  
    CONSTRAINT ...  
);
```

Enforcing Data Integrity

- Entity integrity: Data types, column options, and primary key constraints
- Referential integrity: Foreign key constraints
- **Business rule integrity: Triggers, application-specific**

```
CREATE TABLE tbl_name (  
  col_name data_type [col_opts],  
  col_name ...,  
  CONSTRAINT constraint_name [constraint_options],  
  CONSTRAINT ...  
);
```


Data Types (Entity Integrity)

- A way to enforce domain integrity of attributes.
- Numeric:
 - INT, BIGINT - 32-bit vs. 64-bit, can be unsigned
 - BOOL - TRUE/FALSE values, same as TINYINT(1)
 - FLOAT(M,D), DOUBLE(M,D) - single- vs double-precision floating point (accuracy 7 vs 15 decimal points, depending on platform/implementation)
 - DECIMAL(M,D) - fixed-point number, 65 digit precision

M total number of digits, D number of digits after decimal.

Use DECIMAL instead of FLOAT (or DOUBLE), otherwise you need to use a “tolerance” for inaccuracy.

<http://dev.mysql.com/doc/refman/5.7/en/numeric-type-overview.html>

<http://dev.mysql.com/doc/refman/5.7/en/problems-with-float.html>

Data Types (Entity Integrity)

- String:
 - CHAR(M) - M up to 255, fixed length (right padded on disk)
 - VARCHAR(M) - M up to 64K
 - NCHAR(M), NVARCHAR(M) - UTF8, M up to 64KB^[1]
 - BLOB/TEXT - binary data as byte string (64KB, LONGBLOB 4G)
 - ENUM ('val1','val2',...) - string value must be chosen from list
- Date/Time
 - DATE, TIME, DATETIME -
YYYY-MM-DD HH:MM:SS.fraction
 - TIMESTAMP - 1970-01-01 to 2038-01-19, stored as seconds since unix epoch

1. UTF8 can be 1-3 bytes, depending on the character. Default column limit is 64KB.

<http://dev.mysql.com/doc/refman/5.7/en/string-type-overview.html>

<http://dev.mysql.com/doc/refman/5.7/en/enum.html>

<http://dev.mysql.com/doc/refman/5.7/en/date-and-time-type-overview.html>

<http://dev.mysql.com/doc/refman/5.7/en/storage-requirements.html>

Data Types (Entity Integrity)

```
CREATE TABLE BlogPosts (  
  PostId INT AUTO_INCREMENT,  
  Title VARCHAR(255) NOT NULL,  
  Picture LONGBLOB,  
  Content LONGTEXT,  
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  Published BOOLEAN DEFAULT FALSE,  
  UserName VARCHAR(255),  
  CONSTRAINT pk_BlogPosts_PostId PRIMARY KEY (PostId),  
  CONSTRAINT fk_BlogPosts_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES BlogUsers(UserName)  
    ON UPDATE CASCADE ON DELETE SET NULL  
);
```

Column Options (Entity Integrity)

- NULL/NOT NULL - can be null (missing, unknown value)
- DEFAULT default_value - specifying a default value
- AUTO_INCREMENT [= n] - generate unique id starting at n (surrogate key)

UNIQUE, PRIMARY KEY, FOREIGN KEY REFERENCES tbl_name (col_name,...) can be used as a column option, too. But our convention will be to define them as constraints so they're easier to read.

<http://dev.mysql.com/doc/refman/5.7/en/create-table.html>

<http://dev.mysql.com/doc/refman/5.7/en/example-auto-increment.html>

Column Options (Entity Integrity)

```
CREATE TABLE BlogPosts (  
  PostId INT AUTO_INCREMENT,  
  Title VARCHAR(255) NOT NULL,  
  Picture LONGBLOB,  
  Content LONGTEXT,  
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  Published BOOLEAN DEFAULT FALSE,  
  UserName VARCHAR(255),  
  CONSTRAINT pk_BlogPosts_PostId PRIMARY KEY (PostID),  
  CONSTRAINT fk_BlogPosts_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES BlogUsers(UserName)  
    ON UPDATE CASCADE ON DELETE SET NULL  
);
```

Constraints (Entity Integrity)

- `CONSTRAINT pk_name PRIMARY KEY (col_name,col_name,...);`
Recall that primary keys are unique. Cannot be NULL. Table may only have 1 PK.
- `CONSTRAINT uq_name UNIQUE (col_name,col_name,...);`
EG the alternate candidate key. Can be NULL. Table may contain multiple UQs.

Constraints (Entity Integrity)

```
CREATE TABLE BlogPosts (  
  PostId INT AUTO_INCREMENT,  
  Title VARCHAR(255) NOT NULL,  
  Picture LONGBLOB,  
  Content LONGTEXT,  
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  Published BOOLEAN DEFAULT FALSE,  
  UserName VARCHAR(255),  
  CONSTRAINT pk_BlogPosts_PostId PRIMARY KEY (PostId),  
  CONSTRAINT fk_BlogPosts_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES BlogUsers(UserName)  
    ON UPDATE CASCADE ON DELETE SET NULL  
);
```

Referential Integrity

- Recall normalization can reorganize/decompose tables. Foreign keys allow original tables to be reconstructed (Heath's Theorem).
- Referential integrity: every value of a column in the referencing child table, its foreign key constraint, exists as a value in the referenced parent table, its primary key constraint.
- Ensures consistency when tables properly normalized (each FK value references a PK value that exists).

Constraints (Referential Integrity)

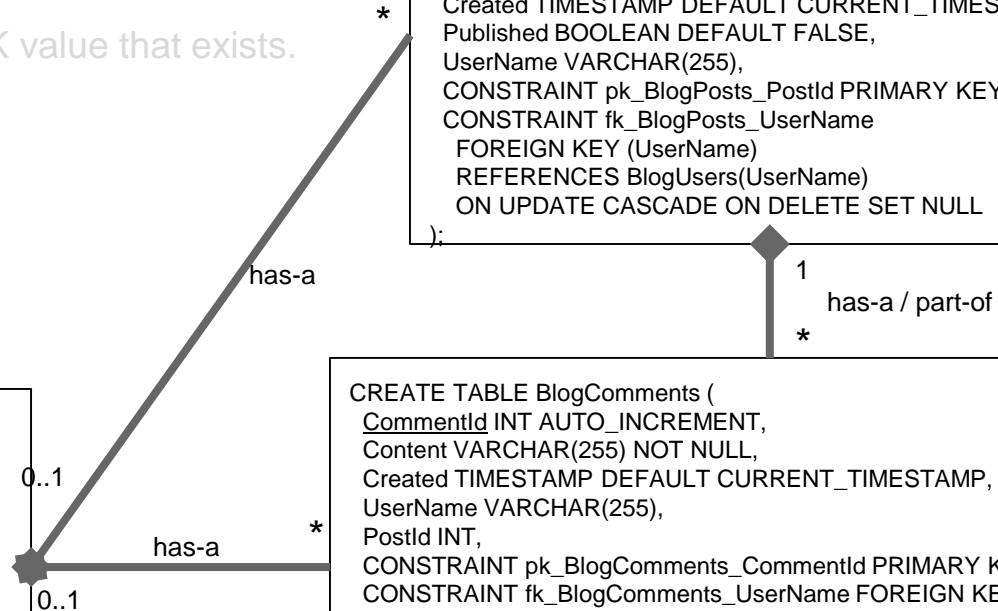
```
CONSTRAINT fk_name FOREIGN KEY (col_name,col_name,...)
REFERENCES tbl_name (col_name,col_name,...)
[ON UPDATE/DELETE reference_opt];
```

Recall each FK value must reference a PK value that exists.

```
CREATE TABLE BlogUsers (
  UserName VARCHAR(255),
  DoB TIMESTAMP NOT NULL,
  StatusLevel ENUM ('novice', 'intermediate', 'advanced'),
  CONSTRAINT pk_BlogUsers_UserName
    PRIMARY KEY (UserName),
  CONSTRAINT fk_BlogUsers_UserName
    FOREIGN KEY (UserName)
      REFERENCES Persons(UserName)
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE BlogPosts (
  PostId INT AUTO_INCREMENT,
  Title VARCHAR(255) NOT NULL,
  Picture LONGBLOB,
  Content LONGTEXT,
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  Published BOOLEAN DEFAULT FALSE,
  UserName VARCHAR(255),
  CONSTRAINT pk_BlogPosts_PostId PRIMARY KEY (PostId),
  CONSTRAINT fk_BlogPosts_UserName
    FOREIGN KEY (UserName)
      REFERENCES BlogUsers(UserName)
      ON UPDATE CASCADE ON DELETE SET NULL
);
```

```
CREATE TABLE BlogComments (
  CommentId INT AUTO_INCREMENT,
  Content VARCHAR(255) NOT NULL,
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UserName VARCHAR(255),
  PostId INT,
  CONSTRAINT pk_BlogComments_CommentId PRIMARY KEY (CommentId),
  CONSTRAINT fk_BlogComments_UserName FOREIGN KEY (UserName)
    REFERENCES BlogUsers(UserName)
    ON UPDATE CASCADE ON DELETE SET NULL,
  CONSTRAINT fk_BlogComments_PostId FOREIGN KEY (PostId)
    REFERENCES BlogPosts(PostId)
    ON UPDATE CASCADE ON DELETE CASCADE
);
```



Constraints (Referential Integrity)

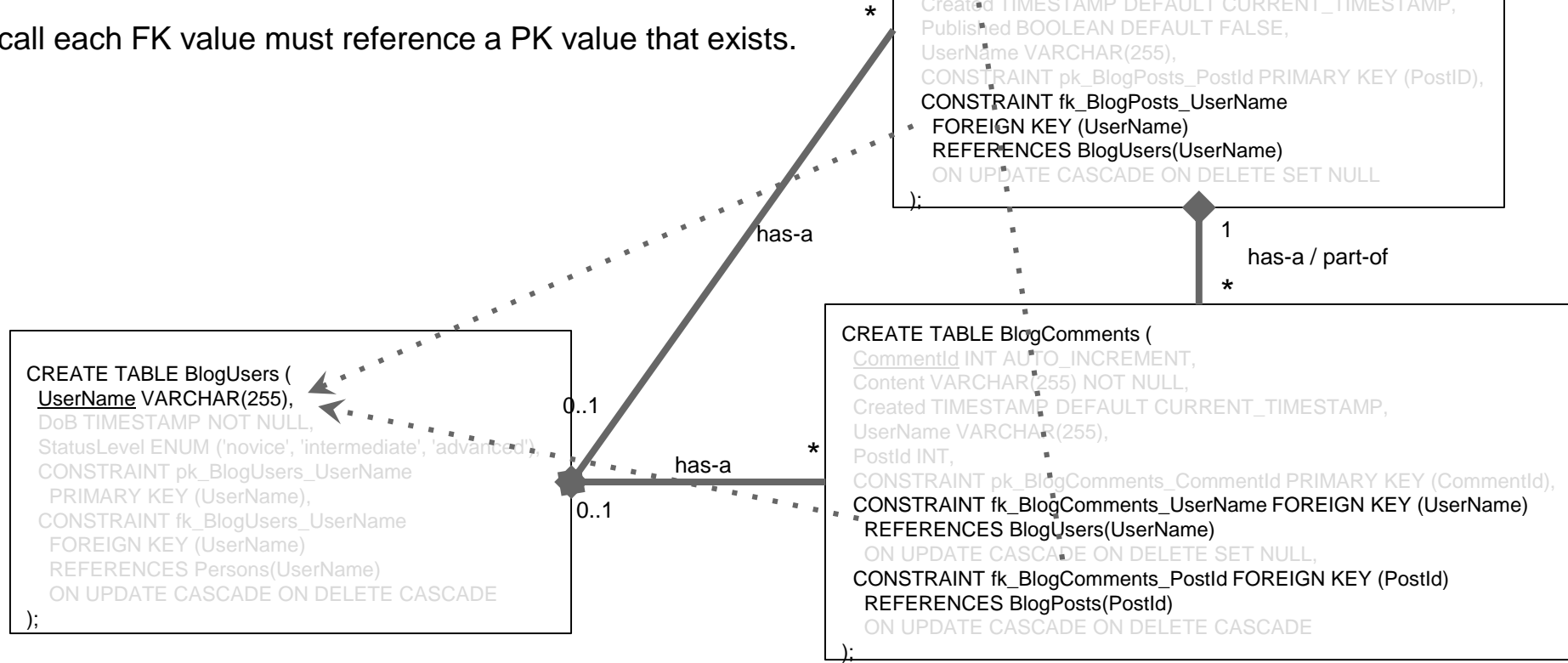
CONSTRAINT fk_name FOREIGN KEY (col_name,col_name,...)
REFERENCES tbl_name (col_name,col_name,...)
[ON UPDATE/DELETE reference_opt];

Recall each FK value must reference a PK value that exists.

```
CREATE TABLE BlogUsers (  
  UserName VARCHAR(255),  
  DoB TIMESTAMP NOT NULL,  
  StatusLevel ENUM ('novice', 'intermediate', 'advanced'),  
  CONSTRAINT pk_BlogUsers_UserName  
    PRIMARY KEY (UserName),  
  CONSTRAINT fk_BlogUsers_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES Persons(UserName)  
    ON UPDATE CASCADE ON DELETE CASCADE  
);
```

```
CREATE TABLE BlogPosts (  
  PostId INT AUTO_INCREMENT,  
  Title VARCHAR(255) NOT NULL,  
  Picture LONGBLOB,  
  Content LONGTEXT,  
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  Published BOOLEAN DEFAULT FALSE,  
  UserName VARCHAR(255),  
  CONSTRAINT pk_BlogPosts_PostId PRIMARY KEY (PostId),  
  CONSTRAINT fk_BlogPosts_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES BlogUsers(UserName)  
    ON UPDATE CASCADE ON DELETE SET NULL  
);
```

```
CREATE TABLE BlogComments (  
  CommentId INT AUTO_INCREMENT,  
  Content VARCHAR(255) NOT NULL,  
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  UserName VARCHAR(255),  
  PostId INT,  
  CONSTRAINT pk_BlogComments_CommentId PRIMARY KEY (CommentId),  
  CONSTRAINT fk_BlogComments_UserName FOREIGN KEY (UserName)  
    REFERENCES BlogUsers(UserName)  
    ON UPDATE CASCADE ON DELETE SET NULL,  
  CONSTRAINT fk_BlogComments_PostId FOREIGN KEY (PostId)  
    REFERENCES BlogPosts(PostId)  
    ON UPDATE CASCADE ON DELETE CASCADE  
);
```



Constraints (Referential Integrity)

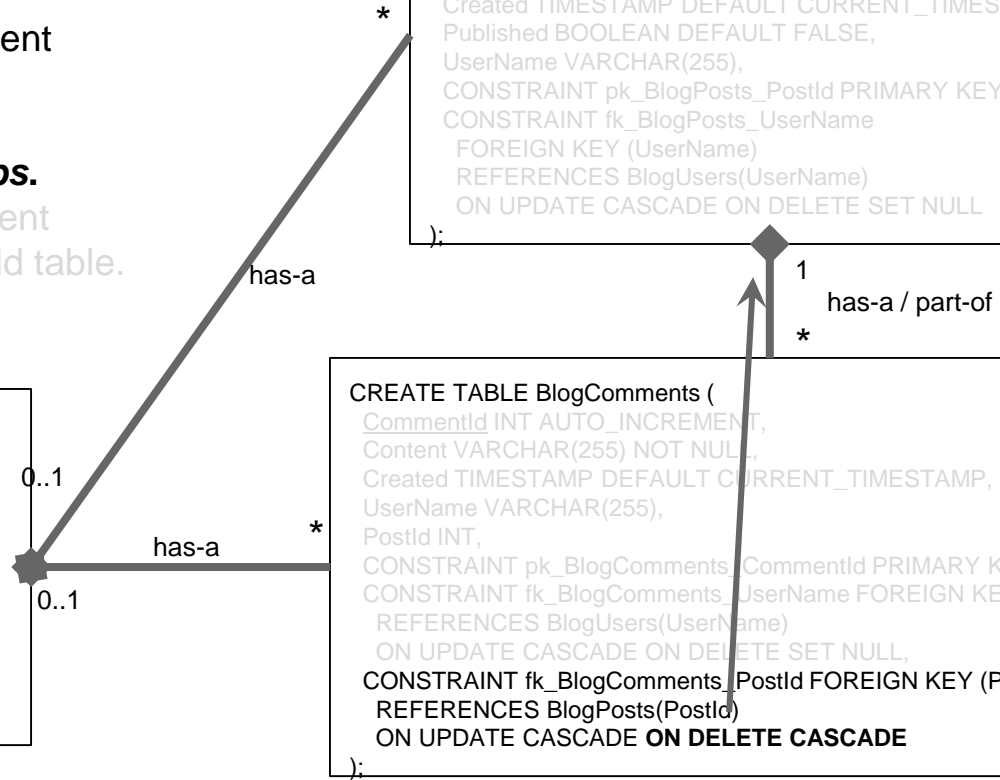
Column Options

- RESTRICT/NO ACTION - default, rejects update/delete in parent table if being referenced in child table.
- CASCADE - for update/delete in parent table, automatically update/delete referencing child table.
Used for composition relationships.
- SET NULL - for update/delete in parent table, set to NULL in referencing child table.
Used for aggregation relationships.

```
CREATE TABLE BlogUsers (  
  UserName VARCHAR(255),  
  DoB TIMESTAMP NOT NULL,  
  StatusLevel ENUM ('novice', 'intermediate', 'advanced'),  
  CONSTRAINT pk_BlogUsers_UserName  
    PRIMARY KEY (UserName),  
  CONSTRAINT fk_BlogUsers_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES Persons(UserName)  
    ON UPDATE CASCADE ON DELETE CASCADE  
);
```

```
CREATE TABLE BlogPosts (  
  PostId INT AUTO_INCREMENT,  
  Title VARCHAR(255) NOT NULL,  
  Picture LONGBLOB,  
  Content LONGTEXT,  
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  Published BOOLEAN DEFAULT FALSE,  
  UserName VARCHAR(255),  
  CONSTRAINT pk_BlogPosts_PostId PRIMARY KEY (PostId),  
  CONSTRAINT fk_BlogPosts_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES BlogUsers(UserName)  
    ON UPDATE CASCADE ON DELETE SET NULL  
);
```

```
CREATE TABLE BlogComments (  
  CommentId INT AUTO_INCREMENT,  
  Content VARCHAR(255) NOT NULL,  
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  UserName VARCHAR(255),  
  PostId INT,  
  CONSTRAINT pk_BlogComments_CommentId PRIMARY KEY (CommentId),  
  CONSTRAINT fk_BlogComments_UserName FOREIGN KEY (UserName)  
    REFERENCES BlogUsers(UserName)  
    ON UPDATE CASCADE ON DELETE SET NULL,  
  CONSTRAINT fk_BlogComments_PostId FOREIGN KEY (PostId)  
    REFERENCES BlogPosts(PostId)  
    ON UPDATE CASCADE ON DELETE CASCADE  
);
```



Constraints (Referential Integrity)

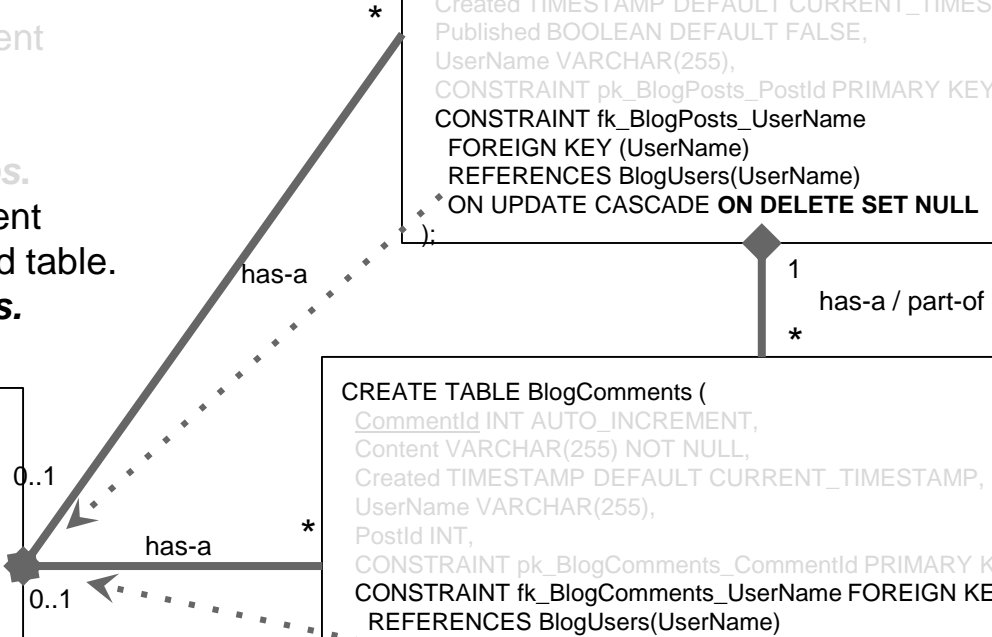
Column Options

- RESTRICT/NO ACTION - default, rejects update/delete in parent table if being referenced in child table.
- CASCADE - for update/delete in parent table, automatically update/delete referencing child table.
Used for composition relationships.
- SET NULL - for update/delete in parent table, set to NULL in referencing child table.
Used for aggregation relationships.

```
CREATE TABLE BlogUsers (  
  Username VARCHAR(255),  
  DoB TIMESTAMP NOT NULL,  
  StatusLevel ENUM ('novice', 'intermediate', 'advanced'),  
  CONSTRAINT pk_BlogUsers_Username  
    PRIMARY KEY (Username),  
  CONSTRAINT fk_BlogUsers_Username  
    FOREIGN KEY (Username)  
      REFERENCES Persons(Username)  
      ON UPDATE CASCADE ON DELETE CASCADE  
);
```

```
CREATE TABLE BlogPosts (  
  PostId INT AUTO_INCREMENT,  
  Title VARCHAR(255) NOT NULL,  
  Picture LONGBLOB,  
  Content LONGTEXT,  
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  Published BOOLEAN DEFAULT FALSE,  
  Username VARCHAR(255),  
  CONSTRAINT pk_BlogPosts_PostId PRIMARY KEY (PostId),  
  CONSTRAINT fk_BlogPosts_Username  
    FOREIGN KEY (Username)  
      REFERENCES BlogUsers(Username)  
      ON UPDATE CASCADE ON DELETE SET NULL  
);
```

```
CREATE TABLE BlogComments (  
  CommentId INT AUTO_INCREMENT,  
  Content VARCHAR(255) NOT NULL,  
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  Username VARCHAR(255),  
  PostId INT,  
  CONSTRAINT pk_BlogComments_CommentId PRIMARY KEY (CommentId),  
  CONSTRAINT fk_BlogComments_Username FOREIGN KEY (Username)  
    REFERENCES BlogUsers(Username)  
    ON UPDATE CASCADE ON DELETE SET NULL,  
  CONSTRAINT fk_BlogComments_PostId FOREIGN KEY (PostId)  
    REFERENCES BlogPosts(PostId)  
    ON UPDATE CASCADE ON DELETE CASCADE  
);
```



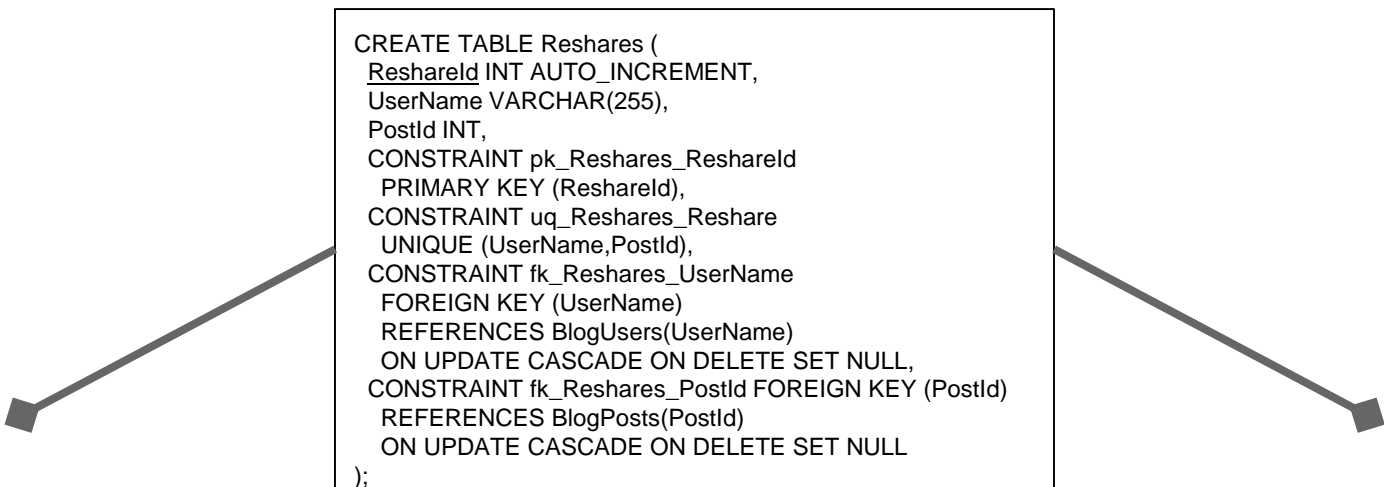
Exercise

- Implement the Reshares table.
Hint: there are two foreign keys, and represent the association relationships with aggregations.

Exercise

- Implement the Reshares table.

Hint: there are two foreign keys, and represent the association relationships with aggregations.



```
CREATE TABLE Reshares (  
  ReshareId INT AUTO_INCREMENT,  
  UserName VARCHAR(255),  
  PostId INT,  
  CONSTRAINT pk_Reshares_ReshareId  
    PRIMARY KEY (ReshareId),  
  CONSTRAINT uq_Reshares_Reshare  
    UNIQUE (UserName,PostId),  
  CONSTRAINT fk_Reshares_UserName  
    FOREIGN KEY (UserName)  
      REFERENCES BlogUsers(UserName)  
      ON UPDATE CASCADE ON DELETE SET NULL,  
  CONSTRAINT fk_Reshares_PostId FOREIGN KEY (PostId)  
    REFERENCES BlogPosts(PostId)  
      ON UPDATE CASCADE ON DELETE SET NULL  
);
```

CREATE TRIGGER (Business Integrity)

- Triggers are a way to enforce business rules.

```
CREATE TRIGGER trigger_name {BEFORE|AFTER}  
  {INSERT|UPDATE|DELETE}  
  ON tbl_name FOR EACH ROW  
  [{FOLLOWS|PRECEDES} other_trigger]  
  BEGIN ... END;
```

BEGIN ... END is a list of statements.

<http://dev.mysql.com/doc/refman/5.7/en/create-trigger.html>

<http://dev.mysql.com/doc/refman/5.7/en/trigger-syntax.html>

<http://dev.mysql.com/doc/refman/5.7/en/stored-program-restrictions.html>

<http://dev.mysql.com/doc/refman/5.7/en/begin-end.html>

DROP TABLE

`DROP TABLE [IF EXISTS] tbl_name1,tbl_name2,...;`

Deletes the table definition and the data. Requires drop privilege.

Useful for recreating tables. For example, include the following before a CREATE TABLE tbl_name statement:

`DROP TABLE IF EXISTS tbl_name;`

<http://dev.mysql.com/doc/refman/5.7/en/drop-table.html>

ALTER TABLE

ALTER TABLE tbl_name alter_spec [, alter_spec, ...];

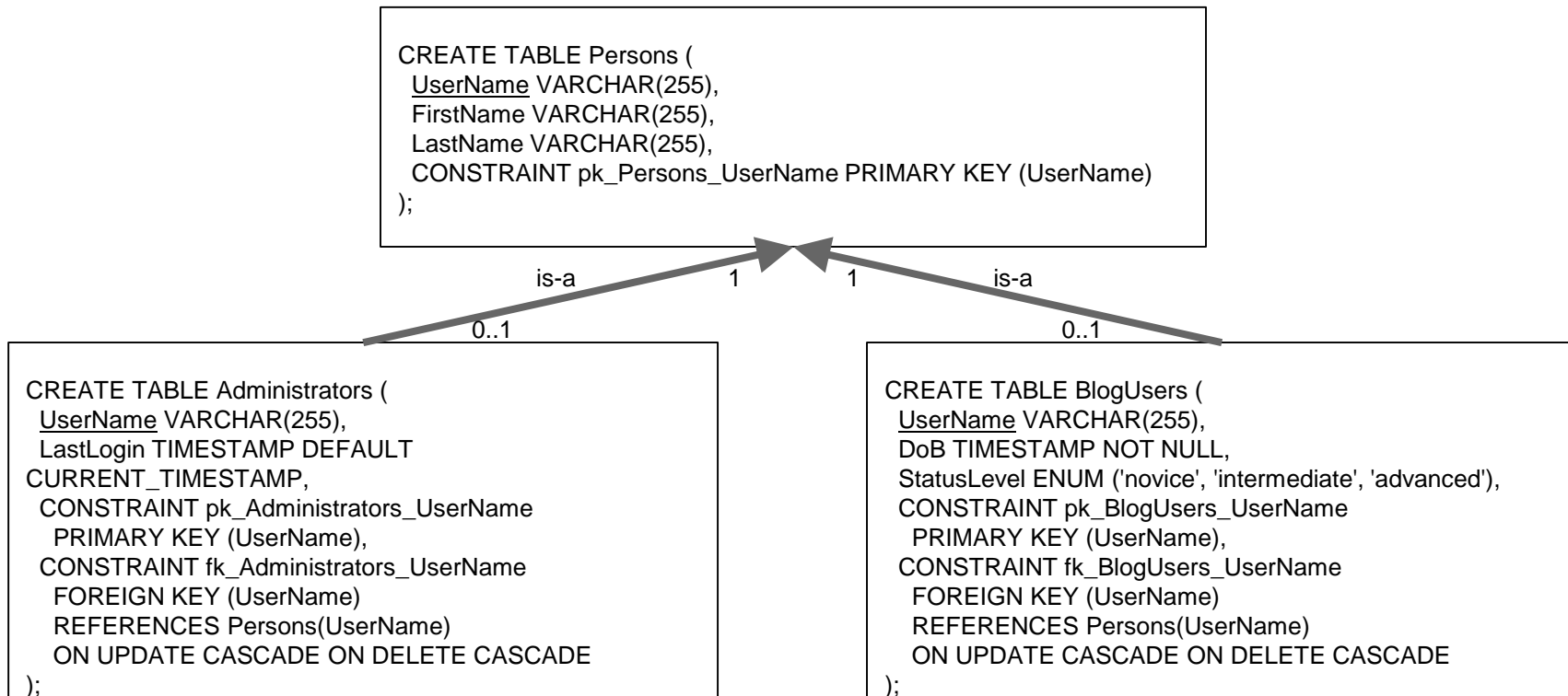
- alter_spec:
 - RENAME AS new_tbl_name
 - ADD COLUMN column_def
 - MODIFY COLUMN col_name column_def
 - DROP {COLUMN col_name|PRIMARY KEY|FOREIGN KEY fk_name}
 - ADD CONSTRAINT constraint_def

Implement a Relational Model

L2: Implement Inheritance

Multi-table Inheritance

- Define a table for the superclass and a table for each subclass.




Multi-table Inheritance

- The primary key for each subclass is also its foreign key.

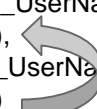
This foreign key references the primary key of the superclass.

```
CREATE TABLE Persons (  
  UserName VARCHAR(255),  
  FirstName VARCHAR(255),  
  LastName VARCHAR(255),  
  CONSTRAINT pk_Persons_UserName PRIMARY KEY (UserName)  
);
```

```
CREATE TABLE Administrators (  
  UserName VARCHAR(255),  
  LastLogin TIMESTAMP DEFAULT  
  CURRENT_TIMESTAMP,  
  CONSTRAINT pk_Administrators_UserName  
    PRIMARY KEY (UserName),  
  CONSTRAINT fk_Administrators_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES Persons(UserName)  
    ON UPDATE CASCADE ON DELETE CASCADE  
);
```



```
CREATE TABLE BlogUsers (  
  UserName VARCHAR(255),  
  DoB TIMESTAMP NOT NULL,  
  StatusLevel ENUM ('novice', 'intermediate', 'advanced'),  
  CONSTRAINT pk_BlogUsers_UserName  
    PRIMARY KEY (UserName),  
  CONSTRAINT fk_BlogUsers_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES Persons(UserName)  
    ON UPDATE CASCADE ON DELETE CASCADE  
);
```



Multi-table Inheritance

- The primary key for each subclass is also its foreign key.
This foreign key references the primary key of the superclass.

```
CREATE TABLE Persons (  
  UserName VARCHAR(255),  
  FirstName VARCHAR(255),  
  LastName VARCHAR(255),  
  CONSTRAINT pk_Persons_UserName PRIMARY KEY (UserName)  
);
```

```
CREATE TABLE Administrators (  
  UserName VARCHAR(255),  
  LastLogin TIMESTAMP DEFAULT  
CURRENT_TIMESTAMP,  
  CONSTRAINT pk_Administrators_UserName  
    PRIMARY KEY (UserName),  
  CONSTRAINT fk_Administrators_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES Persons(UserName)  
    ON UPDATE CASCADE ON DELETE CASCADE  
);
```

```
CREATE TABLE BlogUsers (  
  UserName VARCHAR(255),  
  DoB TIMESTAMP NOT NULL,  
  StatusLevel ENUM ('novice', 'intermediate', 'advanced'),  
  CONSTRAINT pk_BlogUsers_UserName  
    PRIMARY KEY (UserName),  
  CONSTRAINT fk_BlogUsers_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES Persons(UserName)  
    ON UPDATE CASCADE ON DELETE CASCADE  
);
```

Multi-table Inheritance

- Set the reference option to
ON UPDATE CASCADE ON DELETE CASCADE.

Ensures update/delete in superclass propagates to subclass (when a record is deleted from the superclass, then the referencing subclass record is deleted, too).

- Possible limitation: deleting a subclass record does not delete the superclass record.

May or may not be a concern in a given data model -- can the superclass record exist on its own?

```
CREATE TABLE Administrators (  
  UserName VARCHAR(255),  
  LastLogin TIMESTAMP DEFAULT  
  CURRENT_TIMESTAMP,  
  CONSTRAINT pk_Administrators_UserName  
    PRIMARY KEY (UserName),  
  CONSTRAINT fk_Administrators_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES Persons(UserName)  
  ON UPDATE CASCADE ON DELETE CASCADE  
);
```

```
CREATE TABLE BlogUsers (  
  UserName VARCHAR(255),  
  DoB TIMESTAMP NOT NULL,  
  StatusLevel ENUM ('novice', 'intermediate', 'advanced'),  
  CONSTRAINT pk_BlogUsers_UserName  
    PRIMARY KEY (UserName),  
  CONSTRAINT fk_BlogUsers_UserName  
    FOREIGN KEY (UserName)  
    REFERENCES Persons(UserName)  
  ON UPDATE CASCADE ON DELETE CASCADE  
);
```

Single-table Inheritance

- *We will not be using this for our class.*
- The superclass and all its subclasses are modeled by one table definition.

```
CREATE TABLE Persons (  
  UserName VARCHAR(255),  
  FirstName VARCHAR(255),  
  LastName VARCHAR(255),  
  Type ENUM('Administrator','BlogUser')  
  LastLogin TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  DoB TIMESTAMP NOT NULL,  
  StatusLevel ENUM ('novice', 'intermediate', 'advanced'),  
  CONSTRAINT pk_Persons_UserName PRIMARY KEY (UserName)  
);
```

Single-table Inheritance

- Limitation: every record contains attributes for all subclasses.
 - Can break the encapsulation of conceptual object model (each subclass has attributes from other subclasses, which are irrelevant).
 - Potentially violates NF (fields may have a dependency on subclass type).
 - May impact performance (requires more storage).
 - If the only difference is the 'type', then you can use an enum instead.

```
CREATE TABLE Persons (  
  UserName VARCHAR(255),  
  FirstName VARCHAR(255),  
  LastName VARCHAR(255),  
  Type ENUM('Administrator','BlogUser')  
  LastLogin TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  DoB TIMESTAMP NOT NULL,  
  StatusLevel ENUM ('novice', 'intermediate', 'advanced'),  
  CONSTRAINT pk_Persons_UserName PRIMARY KEY (UserName)  
);
```


Implement a Relational Model

L3: Load and Modify Data

INSERT INTO table

```
INSERT INTO tbl_name(col_name,col_name,...)  
VALUES (val1,val2,...);
```

```
INSERT INTO tbl_name(col_name,col_name,...)  
VALUES (val1,val2,...),  
        (val11,val12,...),  
        (val21,val22,...);
```

Alternatively: INSERT INTO table SET

```
INSERT INTO tbl_name  
SET col_name1=val1,col_name2=val2,...;
```

UPDATE table

UPDATE tbl_name

SET col_name1=val1, col_name2=val2, ...

WHERE where_condition;

Example: update my the FirstName of a BlogUser.

More about the “where_condition” in the next lecture, when we talk about SQL. Also the ability to reference multiple tables, for example update all my blog posts to Published=TRUE.

<http://dev.mysql.com/doc/refman/5.7/en/update.html>

DELETE FROM table

```
DELETE FROM tbl_name  
WHERE where_condition;
```

Example: delete blog comments created after a specific time.

More about the “where_condition” in the next lecture, when we talk about SQL. Also the ability to reference multiple tables, for example delete all the comments for my blog posts.

<http://dev.mysql.com/doc/refman/5.7/en/delete.html>

REPLACE INTO table

```
REPLACE INTO tbl_name(col_name,col_name,...)  
VALUES (val1,val2,...);
```

- Same as DELETE and then INSERT. If the PK or UQ exists, then DELETE and INSERT. If not, then just INSERT.

Same syntax as INSERT INTO.

<http://dev.mysql.com/doc/refman/5.7/en/replace.html>

LOAD DATA INFILE

- High speed, bulk data loading
- Example: load CSV with header

```
LOAD DATA INFILE 'data.csv' INTO TABLE tbl_name  
  FIELDS TERMINATED BY ',' ENCLOSED BY '"'  
  LINES TERMINATED BY '\r\n'  
  IGNORE 1 LINES;
```

Note: Mac lines are only terminated with '\n'

- Example: load file and assign value to a column

```
LOAD DATA INFILE 'data.txt'  
  INTO TABLE t1 (col_name1, col_name2)  
  SET col_name3 = CURRENT_TIMESTAMP;
```

Data files can be generated from a DB with a “SELECT ... INTO OUTFILE” statement. More about SELECT statements in the next lecture.

<http://dev.mysql.com/doc/refman/5.7/en/load-data.html>

<http://dev.mysql.com/doc/refman/5.7/en/select-into.html>

Implement a Relational Model

L4: Using MySQL and Workbench

Configuring MySQL & Workbench

- How to start and stop the MySQL server

- Configuring your terminal

In my .bashrc:

```
export PATH=/usr/local/mysql/bin/:$PATH
```

```
$ source .bashrc
```

In my .bash_profile, which runs by default for Mac:

```
if [ -f ~/.bashrc ]; then
```

```
    source ~/.bashrc
```

```
fi
```

- Adding a new user. By default, the 'root' user does not have a password.

<https://dev.mysql.com/doc/refman/5.6/en/adding-users.html>

```
$ mysql --user=root mysql
```

```
mysql> CREATE USER 'root2'@'localhost' IDENTIFIED BY 'password';
```

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root2'@'localhost' WITH GRANT OPTION;
```

Or reset password for 'root': <http://dev.mysql.com/doc/refman/5.6/en/resetting-permissions.html>

- Create a Workbench connection
- The default Workbench port 3306 works on Windows, but not Mac... So which port? 3307?

```
$ defaults read /Library/LaunchDaemons/com.oracle.oss.mysql.mysqld.plist
```

Look for --port=port_num

- Create a schema in Workbench for your tables

Creating Tables & Inserting Data

- CREATE TABLE and DROP TABLE statements must respect referential integrity, so requires order.
 - Cannot create a table that has FK constraints referring to nonexistent tables. Cannot drop a table that other FK constraints refer to.
 - In other words: create parent tables first, drop child tables first.
- INSERT INTO and DELETE FROM also must respect referential integrity.
 - Insert parent entities first, delete child entities first.

Exercise

- Run the CREATE TABLE statements from this module.
- INSERT INTO tables with data similar to our blog application example, or make up your own.
- Use a LOAD DATA INFILE command to bulk load into Persons.

Exercise

- Create tables solution: [3.4.1 Example: Create tables]
- Inserting data solution: [3.4.2 Example: Insert Data]