Basic SQL and the Relational Model

Databases

Ira Assent

ira@cs.au.dk

Data-Intensive Systems group, Department of Computer Science, Aarhus University, DK

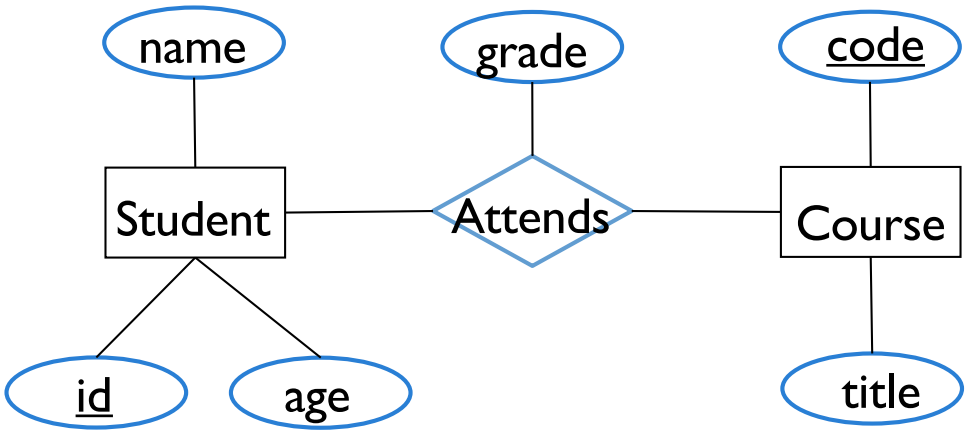
Intended learning outcomes

- Be able to
  - Store data in the relational data model
  - Write basic SQL DDL statements

ira@cs.au.dk

Recap: Design for well working databases

- Design
  - Think before you SQL!
    - Better performance, controlled redundancy, all relevant data, no unnecessary data, data quality



- E/R Diagrams
  - Entity sets, attributes, relationships
  - Cardinality ratios, weak entity sets, subclasses, keys
- Mapping ER diagrams to relational schema
- Design principles: Rules for "good" design

ira@cs.au.dk

Relational Model

- Use a simple data structure: the table
  - simple to understand
  - useful data structure (captures many situations)
    - but richer models, e.g., XML or JSON, are useful in some settings
  - leads to useful yet not too complex query languages

- An elegant mathematical foundation: Relation
  - The contents of the table store the relation
    - So e.g. all the tuples in the Student relation
  - set and multi-set theory
  - relational algebra and calculi

STUDENT

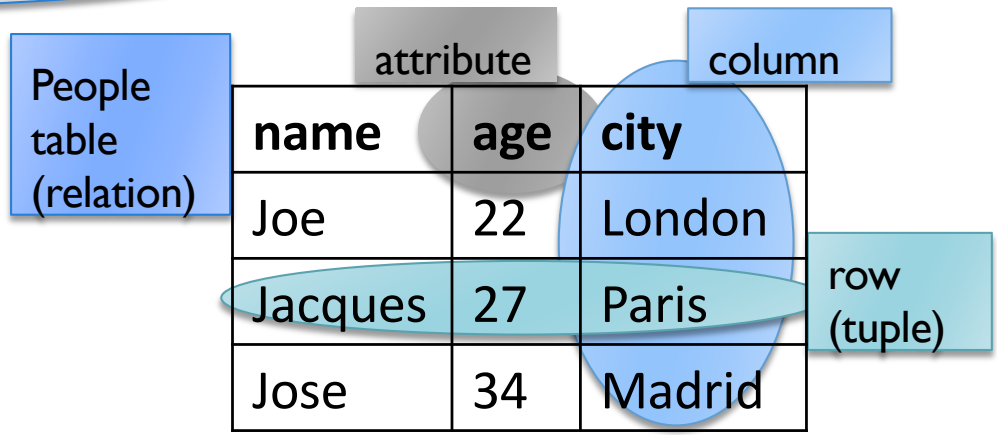
Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

- Allows efficient algorithms
- Industrial strength implementations are available

ira@cs.au.dk

The Relational Data Model

- Data is stored in tables (relations), e.g. People
- Rows and tuples store entries – values for a particular piece of information
  - E.g. all data we have for a particular person in the People table
- Columns contain the name and values for a particular type of information
  - E.g. all the information we have on the cities that people live in
- Attributes are the different types of information in a particular table
  - E.g. cities that people live in
- An attribute value is a piece of information that we store
  - E.g. the city that a particular person lives in



ira@cs.au.dk

Schema

- Schema is the description of the table/relation:
  - People (name, age, city)
- Relation schema:
  - name of the relation,
  - names of the attributes,
  - types of the attributes,
  - constraints

People

schema

name	age	city
Joe	22	London
Jacques	27	Paris
Jose	34	Madrid

- Database schema: collection of all relation schemas

- Abstract tables
  - invariant under permutation of rows and columns
  - no information is stored in the order
- May or may not allow duplicate rows

ira@cs.au.dk

NULL Values

- An attribute value may be NULL
  - it is unknown
  - no value exists
  - it is unknown or does not exist

  - E.g. someone is new to Denmark and does not yet have a CPR number
  - Or, someone has not shared their CPR number yet

- NULL values are treated specially
  - We will see this when we start writing queries

ira@cs.au.dk

T-Rex: NULL here means

1. Both are unknown
2. Both do not exist
3. Color is unknown, zoo does not exist
4. Color does not exist, zoo is unknown

animal	color	zoo
lion	yellow	Copenhagen
crocodile	green	London
Tyrannosaurus Rex	NULL	NULL
polar bear	white	Berlin

ira@cs.au.dk

SQL

- **S**tructured **Q**uery **L**anguage
  - Invented by IBM in the 1970s (many versions, none implements full ANSI standard)
  - **MySQL**, DB2, Oracle, SQL Server, …
- DDL: Data Definition Language for schema definitions
- DML: Data Manipulation Language for data handling

- High-Level, "declarative," no low-level manipulations
- Algebraic foundations, query optimization

ira@cs.au.dk

SQL DDL: Creating Databases

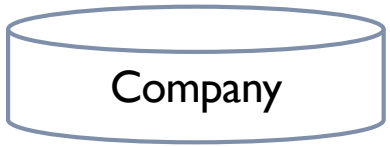
- We start by creating our database

```
CREATE SCHEMA Company;
```

  - In MySQL you can also use

```
CREATE DATABASE Company;
```

  - Identical
  - "Declaration"
    - Makes the name "Company" known, and you can continue defining more of the schema
    - Empty at first



- You get an error message if there is already a database of that name unless you use `IF NOT EXISTS`

```
CREATE {DATABASE | SCHEMA}
[IF NOT EXISTS] db_name;
```

https://dev.mysql.com/doc/refman/8.0/en/create-database.html

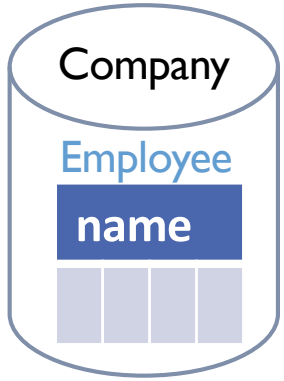
ira@cs.au.dk

SQL DDL: Creating Tables

- We add relations / (base) tables

```
CREATE TABLE Employee (…);
```

  - Creates a table "Employee"
  - But requires at the same time also the definition of the table columns
  - Empty at first



- Column definitions need at least an attribute name and a data type

```
name VARCHAR(40)
```

  - Creates an attribute called "name" that allows us to (*VAR*iably) enter up to 40 *CHAR*acters of text

```
CREATE TABLE Employee (name VARCHAR(40));
```

https://dev.mysql.com/doc/refman/8.0/en/create-table.html

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name (create_definition,...)
create_definition: { col_name column_definition | PRIMARY KEY | UNIQUE | FOREIGN KEY
reference_definition }
column_definition: { data_type [NOT NULL | NULL] [DEFAULT {literal | (expr)} ] }
reference_definition: REFERENCES tbl_name (key_part,...) [ON DELETE reference_option] [ON
UPDATE reference_option]
reference_option: RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```


ira@cs.au.dk

SQL Data Types: Numeric

- Choosing data types
- Based on what kind of data we store in an attribute, we need to pick the right data type
  - Choice important for later manipulation
  - For example, store a number as a numeric data type to support arithmetic (e.g. sum over all employee salaries)
- Numeric data types for numbers

```
INT                 217
```

possible to set the number of digits, e.g., INT(4), the default is INT(11)

```
FLOAT               3.14, 42, 0.0018
```

FLOAT(size,d): maximum number of digits may be specified in size parameter, maximum number of digits to the right of the decimal point in d parameter

- Are important examples for integer values and for real numbers
- Many others exist to allow for different precision etc

https://dev.mysql.com/doc/refman/8.0/en/numeric-types.html

ira@cs.au.dk

SQL Data Types: Character-string

- Character or string data types for short text
- Specify the length n
  - Can be fixed length `CHAR(n)`

    `CHAR(2)` **e.g.** `'aa', 'ab', '12', '++'`
    - If shorter, padded with blank characters to the right
    - Padding ignored in comparison operations
  - Or variable length with maximum length n `VARCHAR(n)`

    `VARCHAR(5)` **e.g.** `'', '12345', 'foo', 'x','y'`

- Pick CHAR when values are expected to be close to same length
  - e.g. CPR numbers
    - bonus question, numeric data type?
  - because VARCHAR introduces a bit of overhead (slightly more storage space)
- else use VARCHAR to avoid wasting space
- Comparison of strings is based on lexicographic order, i.e., making use of alphabetic ordering

https://dev.mysql.com/doc/refman/8.0/en/string-types.html

ira@cs.au.dk

More SQL Data Types

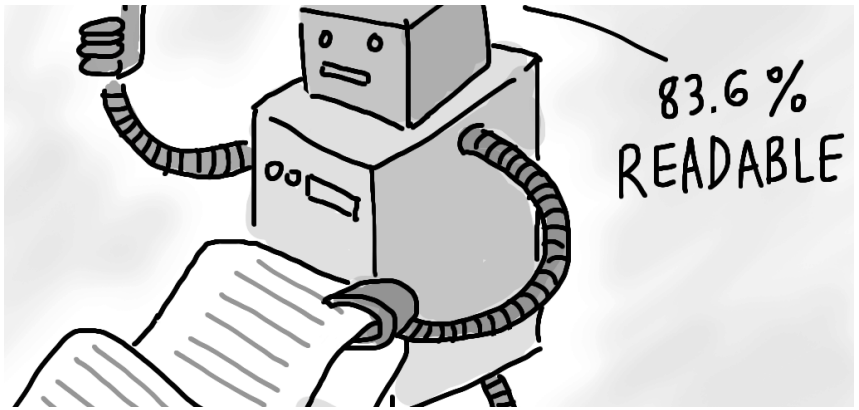
- **Bit-string** data types
  - Fixed length: `BIT(n)`
  - Varying length: `BIT VARYING(n)`
- **Boolean** data type
  - Values of `TRUE` or `FALSE` or `NULL`

- **DATE** data type
  - Ten positions: components `YEAR`, `MONTH`, and `DAY` in the form YYYY-MM-DD
- **TIMESTAMP** data type
  - `DATE` and `TIME` fields plus at least six positions for seconds in decimals, optionally `WITH TIME ZONE` qualifier
  - `TIMESTAMP '2022-02-08 12:35:59.001234'`
- And many more! Additional data types also introduced by different DBMS' – check documentation

https://dev.mysql.com/doc/refman/8.0/en/date-and-time-types.html
https://dev.mysql.com/doc/refman/8.0/en/data-types.html

Attribute Data Types and Domains in SQL

- Domain
  - Name used with the attribute specification
  - Makes it easier to change the data type for a domain that is used by numerous attributes
  - Improves schema readability
  - Example:
    - `CREATE DOMAIN CPR_NR AS CHAR(10);`



ira@cs.au.dk

Which works best?

- CREATE TABLE Student (name, age);
- CREATE TABLE Student (CHAR(30) name, INT age);
- CREATE TABLE Student (name INT, age INT);
- CREATE TABLE Student (name CHAR (30), age INT);

ira@cs.au.dk

Relational Model Constraints

- Constraints
  - Restrictions on the actual values in a database state
  - Derived from the rules in the miniworld that the database represents
  - Domain constraints are one example
    - E.g. DATE only permits values that fit the date type
    - 2024-02-08

ira@cs.au.dk

Relational Database Schema and State

- **Relational database schema** S
  - Set of relation schemas $S = \{R_1, R_2, ..., R_m\}$
  - Set of integrity constraints IC
- **Relational database state**
  - Set of relation states $DB = \{r_1, r_2, ..., r_m\}$
  - Each r_i is a state of R_i and such that the r_i relation states satisfy integrity constraints specified in IC
  - **Invalid state**
    - Does not obey all the integrity constraints
  - **Valid state**
    - Satisfies all the constraints in the defined set of integrity constraints IC

ira@cs.au.dk

Key Constraints

- **Key / superkey**
  - Subset of the attributes of the relation (can contain one attribute, several or all attributes)
  - No two distinct tuples (=rows) have the same value for key
  - E.g. no two students with the same value for {Student_Id}, no two courses with the same value for {title, year}
- **Candidate key**
  - **Minimal superkey**: removing any attribute leaves a set of attributes that is no longer a superkey
    - E.g. {First_name, Last_name, B_day}: if we remove birthday, two students with same name possible, if we remove first name, twins are an issue, and there may be two Emmas born on the same day
  - Relation schema may have more than one candidate key
- **Primary key** of the relation
  - **Designated among candidate keys** (chosen by developer) to identify rows in the table
  - Usually prefer primary key with fewer attributes
  - Mark other candidate keys as unique (coming up in a few slides)

ira@cs.au.dk

Key Constraints textbook example

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

CAR relation, with two candidate keys: License_number and Engine_serial_number

ira@cs.au.dk

Specifying Key and Referential Integrity Constraints

- **`PRIMARY KEY`** clause
  - Specifies one or more attributes that make up the primary key of a relation
  - `Dnumber INT PRIMARY KEY;`

- **`UNIQUE`** clause
  - Specifies alternate (secondary) keys
  - `Dname VARCHAR(15) UNIQUE;`

https://dev.mysql.com/doc/refman/8.0/en/create-table.html

ira@cs.au.dk

Multi-Attribute Key

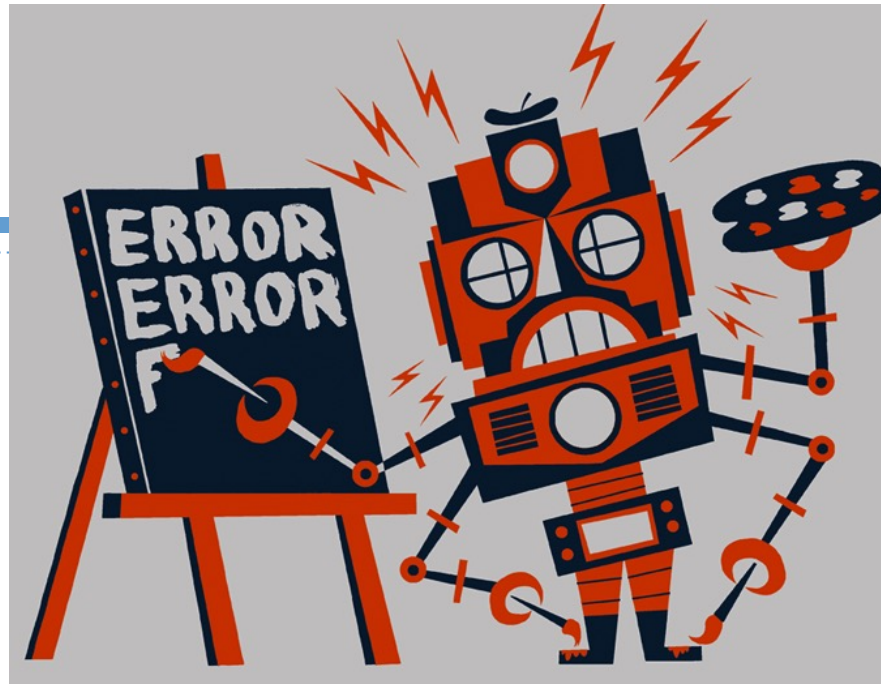
- PRIMARY KEY ($a_1, a_2, ..., a_n$)
- A top-level element of the CREATE statement

```
CREATE TABLE Exams (
    studid CHAR(2),
    date DATE,
    time TIME,
    vip VARCHAR(15),
    room VARCHAR(40),
    PRIMARY KEY (studid, date)
  );
```

studid	date	time	vip	room
01	2014-10-15	09:00	ira	Turing-230
02	2014-10-15	09:30	ira	Turing-230
01	2014-10-16	12:30	amoeller	Turing-230
03	2014-10-16	10:30	bodker	Ada-017
01	2014-10-17	11:00	bodker	Ada-017

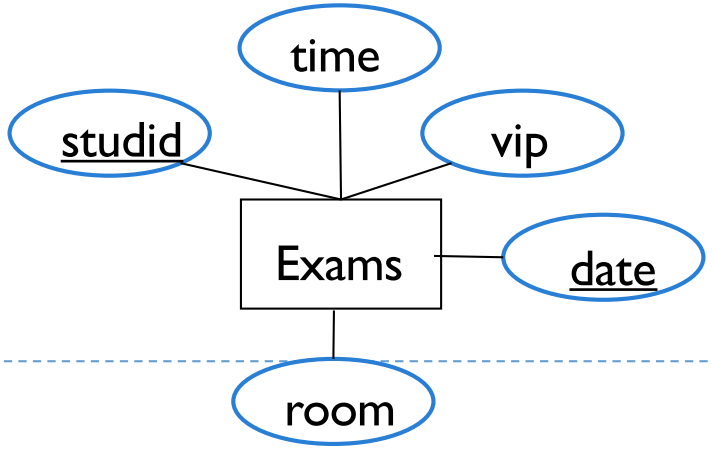
ira@cs.au.dk

Enforcing Key Constraints



- Checked during insert or update
- If it is violated, then a runtime error occurs

```
('01', '2024-06-15', '15:30:00',
'ira', 'Turing-230')
('01', '2024-06-15', '12:45:00',
  'amoeller', 'Ada-017')
```

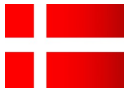



```
CREATE TABLE Exams (
    studid CHAR(2),
    date DATE,
    time TIME,
    vip VARCHAR(15),
    room VARCHAR(40),
    PRIMARY KEY (studid, date)
  );
```

Attribute Constraints and Defaults

- `NOT NULL`
  - `NULL` is not permitted for a particular attribute
- No primary key can be NULL
- Default value
  - **`DEFAULT`** `<value>`

```
CREATE TABLE Employee (
    name VARCHAR(20),
    country VARCHAR (5) DEFAULT 'DK'
);
```



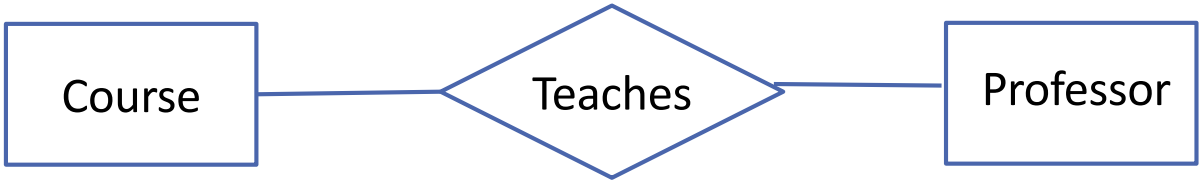
https://dev.mysql.com/doc/refman/8.0/en/create-table.html

ira@cs.au.dk

Foreign Key

- Specifies that an attribute must reference an attribute in another table
  - "**referential integrity**"
- The referenced attribute must be a primary key
- Also possible for multi-attribute keys
- **`FOREIGN KEY`** clause

- Allows representing relationships from E/R diagrams e.g.



https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html

ira@cs.au.dk

Enforcing Foreign Key Constraints

- Course table (**source** / **parent**) references Professor (**target** / **child**)
  - Meaning: tuple in Course has prof_id value that exists in Professor

```
CREATE TABLE Professor (
        id INT PRIMARY KEY,
        name VARCHAR(30) NOT NULL,
        group VARCHAR (15)
);
CREATE TABLE Course (
        code INT PRIMARY KEY,
        name VARCHAR(20) NOT NULL,
        prof_id INT,
        FOREIGN KEY (prof_id) REFERENCES Professor(id)
);
```

Note: allows NULL values for prof_id in Course
i.e., allows a course not to have a teacher assigned
(unlike earlier example with total participation)

- Or inline definition (identical):

```
CREATE TABLE Course (
        code INT PRIMARY KEY,
        name VARCHAR(20) NOT NULL,
        prof_id INT REFERENCES Professor(id));
```


ira@cs.au.dk

SQL CREATE TABLE for COMPANY schema

```
CREATE TABLE EMPLOYEE
        ( Fname                 VARCHAR(15)         NOT NULL,
          Minit                 CHAR,
          Lname                 VARCHAR(15)         NOT NULL,
          Ssn                   CHAR(9)             NOT NULL,
          Bdate                 DATE,
          Address               VARCHAR(30),
          Sex                   CHAR,
          Salary                DECIMAL(10,2),
          Super_ssn             CHAR(9),
          Dno                   INT                 NOT NULL,
        PRIMARY KEY (Ssn),
CREATE TABLE DEPARTMENT
        ( Dname                 VARCHAR(15)         NOT NULL,
          Dnumber               INT                 NOT NULL,
          Mgr_ssn               CHAR(9)             NOT NULL,
          Mgr_start_date        DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber               INT                 NOT NULL,
          Dlocation             VARCHAR(15)         NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```


ira@cs.au.dk

Foreign key constraints

```
CREATE TABLE Rooms(
  room VARCHAR(15)PRIMARY KEY,capacity INT   );
CREATE TABLE People(
  name VARCHAR(40) NOT NULL,
  office VARCHAR(15)REFERENCES Rooms(room),
  userid VARCHAR(15) PRIMARY KEY              );
```

1. A tuple in People has a pointer to a tuple in Rooms.
2. A tuple in Rooms has a pointer to a tuple in People.
3. A tuple in People has an office value that exists in Rooms.
4. A tuple in Rooms has a room value that exists in People.
5. A tuple in People is repeated in Rooms.
6. A tuple in Rooms is repeated in People.

ira@cs.au.dk

Violated Referential Integrity Constraints

- Source or target table may change
  - Spurious value due to insert or update in source
  - Dangling value due to delete or update in target
  - default: reject the insert, delete or update
- Specify behavior with **referential triggered action** clause
  - `CASCADE`: make the same change in the source
    - Suitable for "relationship" constraints
    - E.g. if course code changes in Course propagate change to Student
  - `SET NULL`: change source value to NULL
    - E.g. if textbook goes out of print (deleted from Textbook), remove it from Course by setting corresponding textbook values to NULL
  - `SET DEFAULT`: change source value to DEFAULT
    - E.g. if a Brightspace theme is removed from Themes, set preferred theme in User to DEFAULT *PlainWeird*

ira@cs.au.dk

Foreign Key Constraints Triggered Actions

- Course table (**source** / **parent**) references Professor (**target** / **child**)
  - Meaning: tuple in Course has prof_id value that exists in Professor

```
CREATE TABLE Professor (
        id INT PRIMARY KEY,
        name VARCHAR(30) NOT NULL,
        group VARCHAR (15)
);
CREATE TABLE Course (
        code INT PRIMARY KEY,
        name VARCHAR(20) NOT NULL,
        prof_id INT,
        FOREIGN KEY (prof_id) REFERENCES
        Professor(id) ON DELETE SET NULL ON UPDATE CASCADE
);
```

Note: allows NULL values for prof_id in Course i.e., allows a course not to have a teacher assigned (unlike earlier example with total participation)

ira@cs.au.dk

Example Actions

Rooms (target)

room	capacity
Ny-357	6
Ada-333	26
Ho-017	4

room	capacity
Ny-HQ	6
Ada-333	26

```
CREATE TABLE People (  name
VARCHAR(40) NOT NULL,
   office VARCHAR(15) REFERENCES
   Rooms(room)
   ON DELETE SET ? ON UPDATE ?,
   userid     VARCHAR(15) PRIMARY KEY,
   `group`    CHAR(3)   );
```

People (source)

userid	name	group	office
ira	Ira Assent	vip	Ny-357
aas	Annika Schmidt	phd	NULL
jan	Jan Christensen	tap	Ho-017

? Cascade

userid	name	group	office
ira	Ira Assent	vip	Ny-HQ
aas	Annika Schmidt	phd	NULL

? Set NULL

userid	name	group	office
ira	Ira Assent	vip	NULL
aas	Annika Schmidt	phd	NULL
jan	Jan Christensen	tap	NULL

- Delete Ho-017 from rooms
- Call Ny-357 Ny-HQ instead (update)

Default values, referential integrity actions

```
CREATE TABLE EMPLOYEE
    ( … ,
      Dno          INT          NOT NULL       DEFAULT 1,
    CONSTRAINT EMPPK
      PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
      FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
                    ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
      FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                    ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPARTMENT
    ( … ,
      Mgr_ssn CHAR(9)          NOT NULL       DEFAULT '888665555',
      … ,
    CONSTRAINT DEPTPK
      PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
      UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
      FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                    ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPT_LOCATIONS
    ( … ,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
                    ON DELETE CASCADE           ON UPDATE CASCADE);
```


ira@cs.au.dk

Attribute Constraints

- `CHECK` clauses at the end of a `CREATE TABLE` statement
  - Apply to each tuple individually
  - `CHECK (Dept_create_date <= Mgr_start_date);`
  - `Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);`
- CHECK (*condition*) after an attribute
- Allows any Boolean (evaluates to true or false) including SQL subqueries (introduced later)

```
CREATE TABLE People(name VARCHAR(40)NOT NULL,
       office VARCHAR(15), userid VARCHAR(15)
       PRIMARY KEY,`group` CHAR(3)
       CHECK (`group`='vip' OR `group`='phd'
                     OR`group`='tap'));
```

https://dev.mysql.com/doc/refman/8.0/en/create-table.html
https://dev.mysql.com/doc/refman/8.0/en/create-table-check-constraints.html

ira@cs.au.dk

Row Constraints

- CHECK (*condition*) at top-level (end of table definition)
- Conditions are checked during insert or update of *any* attribute

```
CREATE TABLE Products (
    id INT,
    name VARCHAR(15),
    sales_price DECIMAL(10,2) CHECK (sales_price>0),
    purchase_price DECIMAL(10,2)
    CHECK (sales_price - purchase_price>25)
);
```

Attribute constraint

Row constraint

ira@cs.au.dk

Enforcing Attribute Constraints

- Conditions are checked during insert or update of the attribute, but *not* for other modifications
- Thus, foreign keys *cannot* be enforced:

```
CREATE TABLE Rooms(room VARCHAR(15)PRIMARY KEY, capacity INT);
CREATE TABLE People (
  name      VARCHAR(40) NOT NULL,
  office    VARCHAR(15) CHECK (office IN (
  SELECT room FROM Rooms)),
  userid    VARCHAR(15) PRIMARY KEY  );
```

```
CREATE TABLE People(
  name VARCHAR(40) NOT NULL,
  office VARCHAR(15) REFERENCES Rooms(room),
  userid VARCHAR(15) PRIMARY KEY  );
```

ira@cs.au.dk

Which of these is correct?

1. CREATE TABLE Sells (
   shop, product VARCHAR(20) PRIMARY KEY,
   price REAL
   );
2. CREATE TABLE Customers (
   name CHAR(30) PRIMARY KEY,
   age INT DEFAULT 'two');
3. CREATE TABLE Product (
   productid INT PRIMARY KEY, price
   REAL);
4. CREATE TABLE Email (
   cpr CHAR(6) PRIMARY KEY,
   email INT);

ira@cs.au.dk

Summary

- Intended learning outcomes
- Be able to
  - Store data in the relational data model
  - Write basic SQL DDL statements

ira@cs.au.dk

Where to go from here?

- So, we know how to create a database, and put constraints in place
- And, how do we put data in? delete it?
- How do we get data out again?
- Manipulate the schema later?

ira@cs.au.dk

What was this all about?

Guidelines for your own review of today's session

- The relational model consists of…
- A schema contains…
- A standard SQL DDL statement for creating a database with tables is…
  - We can additionally specify…
    - Primary keys are…
      - Other forms of keys…
      - Foreign keys are
    - In SQL they are defined using…
  - When the data changes, integrity constraints may be…
    - We can control how the DBMS should…
      - The following options exist…

ira@cs.au.dk