CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

SQL, DDL, and Mapping E(E)R to Databases

LECTURE 7

Dr. Philipp Leitner

philipp.leitner@chalmers.se

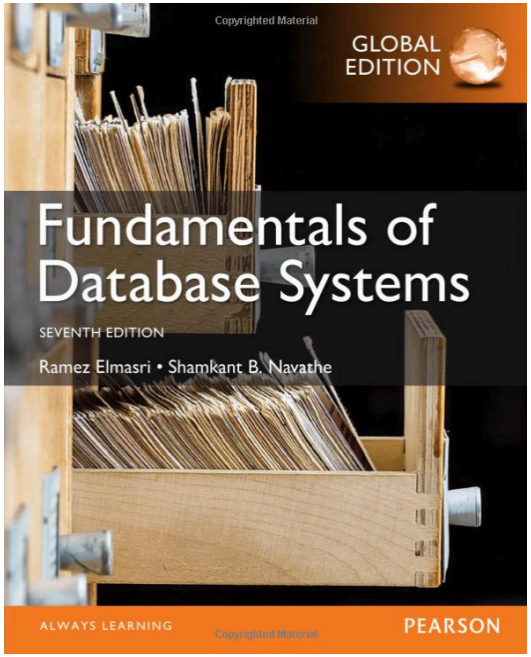
@xLeitix

LECTURE 7

Covers …

Parts of Chapter 6 and 7
Chapter 9

Please read this up until next lecture!



What we will be covering

We finally get started with some SQL

- **Defining tables, inserting data, altering tables**

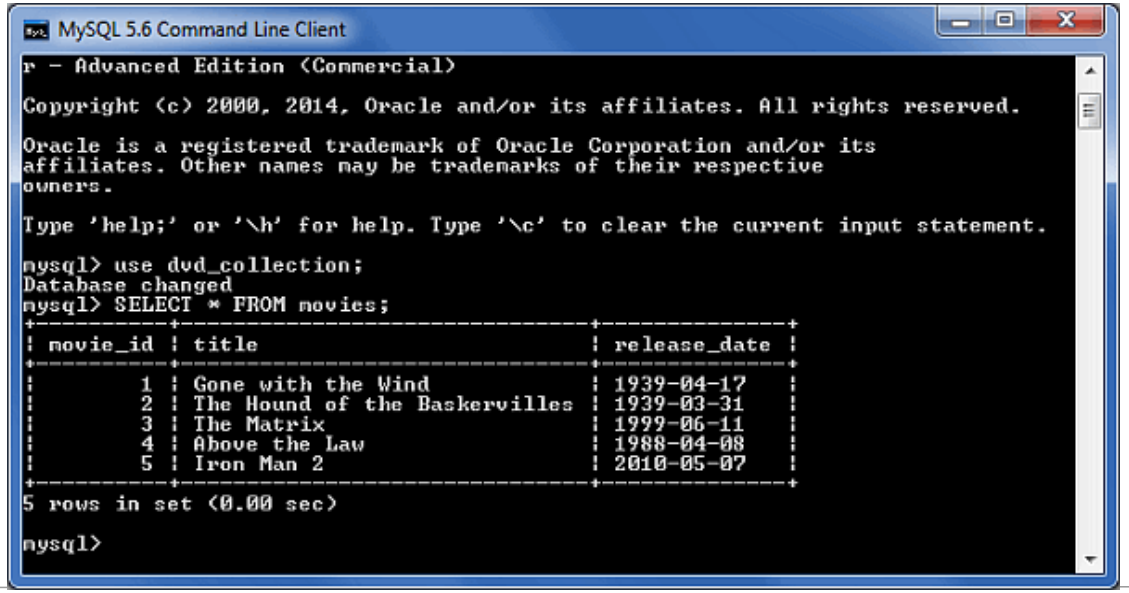
Some patterns for mapping E(E)R to relations and databases

Reminder: DBMS

A database, and a DBMS, is basically a server process

SQL is the **interface** we largely use to interact with this

Conveniently, SQL follows RA **very closely**



SQL

SQL (pronounced “sequel” or sometimes “S-Q-L”)

- Is a standardized **computer language** for interacting with relational databases

ISO/ANSI standard:

- **Standardized** parts work for virtually all relational databases
- Most have also some “non-standard” features (we won’t be covering them)

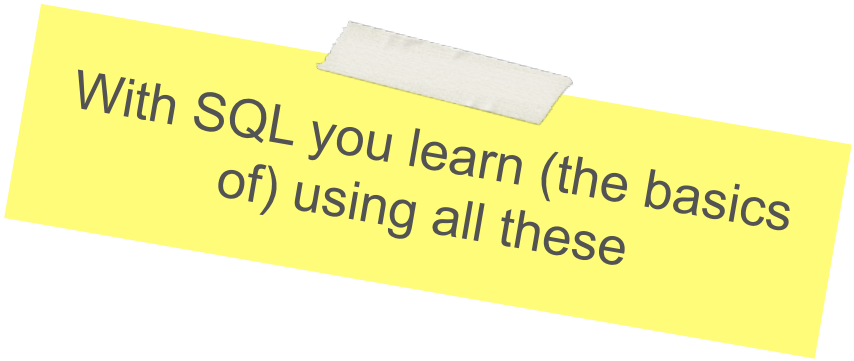
Relational Database Systems

Open Source:

- MySQL
- **PostgreSQL**
- …

Commercial:

- Oracle
- SQL Server (Microsoft)
- …



SQL in a Nutshell

Textual **domain-specific language**

A computer language like Java, but for **two** very specific purposes

Defining a database **schema**

DDL (database definition language)

Inserting, updating, and finding **data items**

DML (data manipulation language)

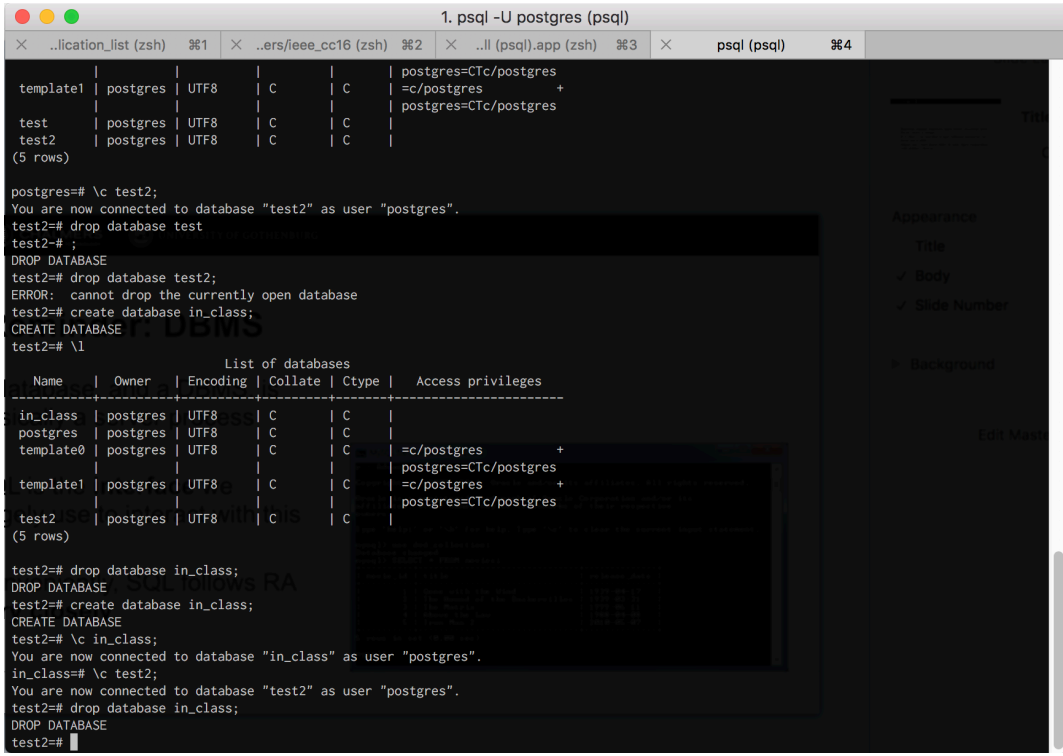
Creating Tables

Basic DDL command is `CREATE TABLE`

Two versions:

- `CREATE TABLE` (base relations)
  - Relation and its tuples are actually created and stored as a file by the DBMS
- `CREATE VIEW` (create virtual relation)
  - Do not correspond to any physical file.

Let’s do a small in-class example. Follow along on your computer if you can.



Getting Started

- Log into your psql client (run '`psql -U postgres`' or something similar in a terminal)
- Type '`\l`' (show all databases)
- Create a database: type '`create database in_class;`'. Validate that the database is created.
- Connect to database: type '`\c in_class;`'

Simple Example

```
CREATE TABLE student (
   personnr CHAR(10),
   name VARCHAR,
   birthdate DATE,
   PRIMARY KEY (personnr)
 );
```

Validating that a table has been created

```
\dt;
```

(prints all tables in the database)

```
SELECT * FROM STUDENT;
```

(shows all content of a table, should currently be empy)

```
\d+ STUDENT;
```

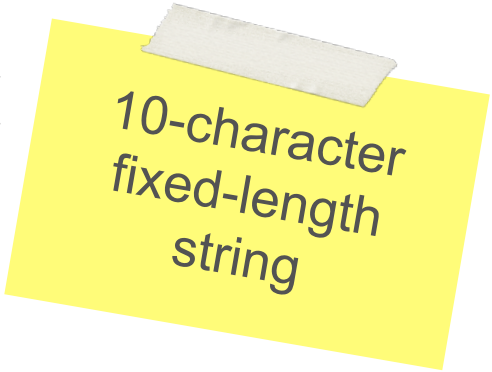
(show table schema)

More info


```
CREATE TABLE STUDENT (
   personnr CHAR(10),
   name VARCHAR,
   birthdate DATE,
   PRIMARY KEY (personnr)
);
```

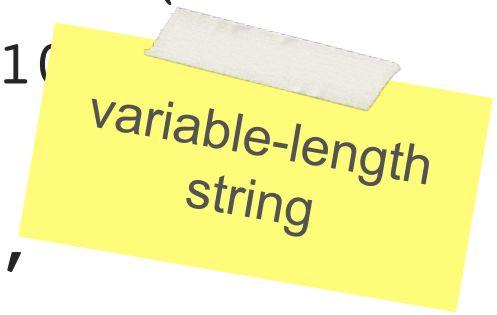
More info

```
CREATE TABLE STUDENT (
   personnr CHAR(10),
   name VARCHAR,
   birthdate DATE,
   PRIMARY KEY (personnr)
);
```



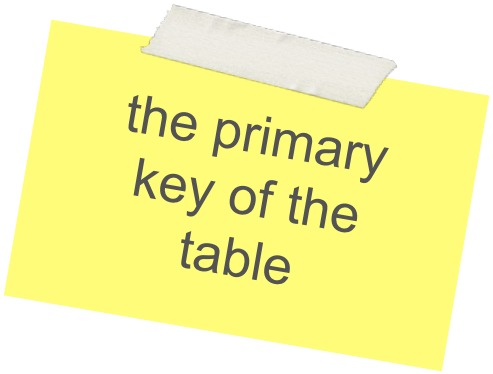
More info

```
CREATE TABLE STUDENT (
    personnr CHAR(10
    name VARCHAR,
    birthdate DATE,
    PRIMARY KEY (personnr)
);
```



More info

```
CREATE TABLE STUDENT (
    personnr CHAR(10),
    name VARCHAR,
    birthdate DATE,
    PRIMARY KEY (personnr)
);
```



Second Example

```
CREATE TABLE ENROLLMENT (
    id INTEGER PRIMARY KEY,
    coursename VARCHAR,
    student CHAR(10),

    FOREIGN KEY (student)
      REFERENCES STUDENT(personnr)
      ON DELETE CASCADE
 );
```

More info

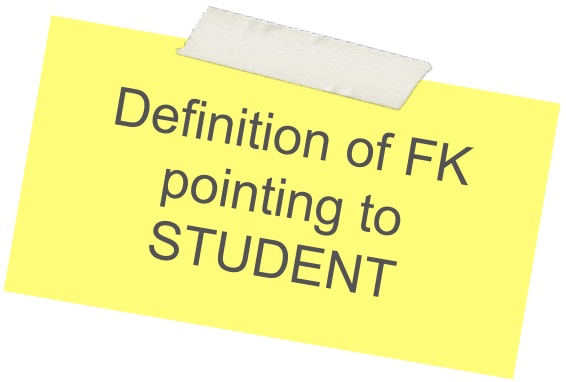
```
CREATE TABLE ENROLLMENT (
    id INTEGER PRIMARY KEY,
    coursename VARCHAR,
    student CHAR(10),

    FOREIGN KEY (student)
      REFERENCES STUDENT(personnr)
      ON DELETE CASCADE
 );
```

More info

```
CREATE TABLE ENROLLMENT (
    id INTEGER PRIMARY KEY,
    coursename VARCHAR,
    student CHAR(10),

    FOREIGN KEY (student)
      REFERENCES STUDENT(personnr)
      ON DELETE CASCADE
 );
```



More info

```
CREATE TABLE ENROLLMENT (
    id INTEGER PRIMARY KEY,
    coursename VARCHAR,
    student CHAR(10),

    FOREIGN KEY (student)
      REFERENCES STUDENT(personnr)
      ON DELETE CASCADE
 );
```

If students are deleted, their enrollment records shall also be deleted (more on this later)

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Copyright (c) 2011 Pearson Education

CREATE TABLE statements to generate these tables

```
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) );
```

Copyright (c) 2011 Pearson Education

Basic Structure

```
CREATE TABLE <TABLE_NAME> (
    ATT1 DOM1 [NOT NULL] [CHECK (<STMT>)],
    ATT2 DOM2 [NOT NULL],
    ATT3 DOM3 [NOT NULL],
    …
    PRIMARY KEY (ATTS),
    [UNIQUE(ATTS),]
    [FOREIGN KEY(ATT3) REFERENCES TABLE2(ATTF)
      [ON [DELETE,UPDATE]
        [NO ACTION, RESTRICT, CASCADE, SET NULL, SET DEFAULT]]]
);
```

2 Basic Observations:

(1) SQL *does not care* about capitalization of keywords

CREATE == create

(2) Statements end with ‘;’

Attribute Data Types

Basic data types

- Numeric data types
  - Integer numbers: `INTEGER` and `SMALLINT`
  - Floating-point (real) numbers: `FLOAT` and `DOUBLE PRECISION`
- Character-string data types
  - Fixed length: `CHAR(n)`
  - Varying length: `VARCHAR(n)`

Attribute 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

Format YYYY-MM-DD

BLOB / CLOB data type

[Binary | Character] Large Objects

E.g., binary or text files

User-Defined Domains

Example:

```
CREATE DOMAIN SSN_TYPE AS CHAR(8);
```

Useful if the same types are used in multiple places in the database

Makes it easier to change them

Constraints

The fundamental ones:

Generic key constraint `UNIQUE(ATTS)`

PK constraints `PRIMARY KEY(ATTS)` (only one per table)

FK constraint

```
FOREIGN KEY(ATT3) REFERENCES TABLE2(ATTF)
```

Constraints

Other constraints

```
DEFAULT <value>
NOT NULL
CHECK (implements arbitrary constraints)
```

Example:

```
Dnumber INT NOT NULL CHECK (
    Dnumber > 0 AND Dnumber < 21);
```

Constraints

Multi-Attribute Primary Key

(has to be written after all attribute definitions)

Example:

```
PRIMARY KEY (essn,dependent_name);
```

Constraints

Constraints can either be written

(1) **directly** when defining the attribute

```
Lname VARCHAR(20) NOT NULL
```

(2) or in a list **at the end** of the `CREATE TABLE` statement

```
CREATE TABLE EMPLOYEE(
    Lname VARCHAR(20),
    …
    NOT NULL(Lname)
)
```

Naming Constraints

Constraints can be explicitly named

May be useful to later comprehend them

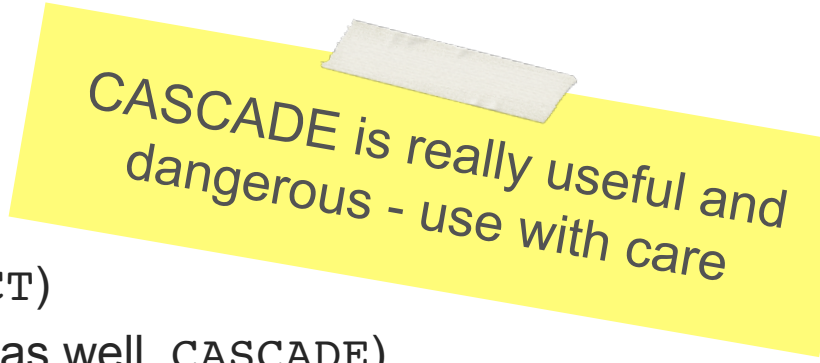
```
CONSTRAINT LnameMustExist NOT NULL(Lname)
```

Referential Integrity Triggers

For `FOREIGN KEY` we can tell the database what it should do with referential integrity violations

A primary key in a **referenced table** changes, what should happen with the **referencing foreign key**?

Can happen through updates or deletes



Four possibilities:

Reject update (default, `NO ACTION, RESTRICT`)

Cascade change (e.g., delete referencing rows as well, `CASCADE`)

Set to NULL or a different default (`SET NULL, SET DEFAULT`)

Changing Table Schemas

After a table is created, it can still be updated (`ALTER TABLE`) or deleted (`DROP TABLE`)

```
DROP TABLE TABLE1;
```

- Deletes `TABLE1` and all data in the table
- Cascades to other tables if configured

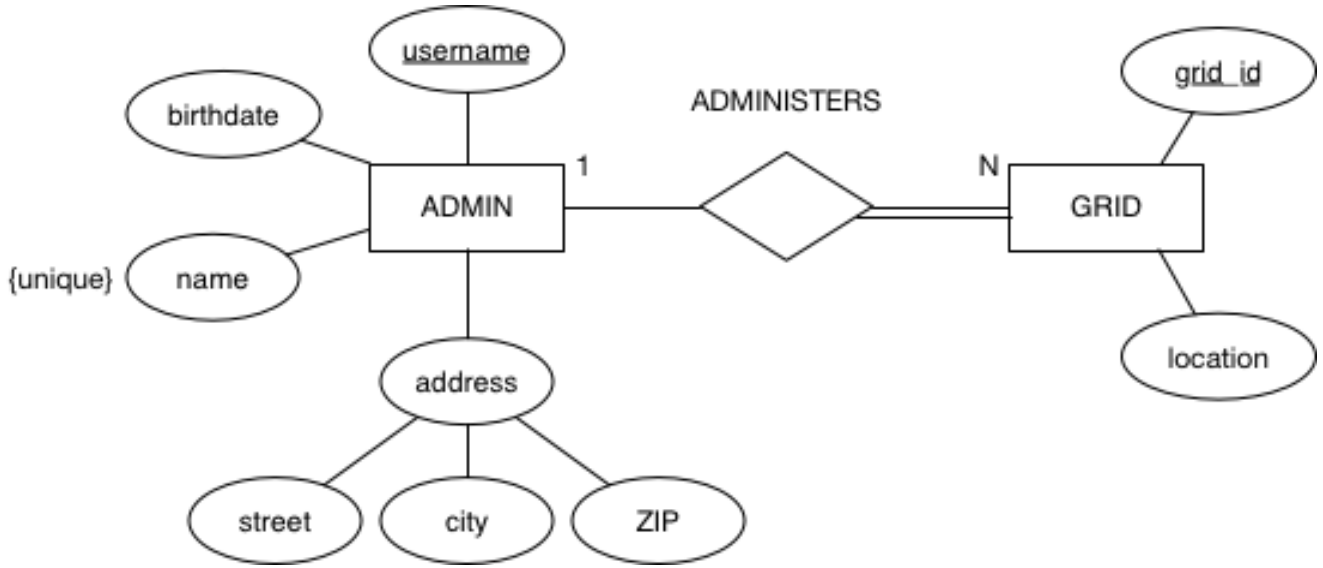
Altering Schemas

Altering tables has a weird syntax. We do not focus on it in this course.

```
ALTER TABLE EMPLOYEE
    DROP COLUMN Address CASCADE
    ADD COLUMN NewAddress VARCHAR(30)
    ALTER COLUMN Fname SET DEFAULT ‘Johan’;
```

In-Class Exercise

Try to create database tables that implement the following simple EER diagram:



SQL DML - Data Manipulation Language

Create
Read
Update
Delete

Create, Updating, and Deleting

Three commands used to modify the database:

`INSERT, DELETE,` and `UPDATE`

`INSERT` inserts a new row into a table

`DELETE` removes or or more rows (that match a condition)

`UPDATE` changes one or more values in all rows that match a condition

We are going to cover the **read** part (querying) extensively in the next lecture.

INSERT - two canonical forms

```
INSERT INTO EMPLOYEE VALUES (
   'Hugo', 'Chavez', '324355423', '1962-12-30',
   'Some Address', 'M', 37000, '325234523', 4
 );
```

Or:

```
INSERT INTO EMPLOYEE(Fname, Lname, Ssn) VALUES (
   'Hugo', 'Chavez', '324323');
```

Not provided values in the second form become NULL

INSERT - two canonical forms

```
INSERT INTO EMPLOYEE VALUES (
   'Hugo', 'Chavez', '324355423', '1962-12-30',
   'Some Address', 'M', 37000, '325234523', 4
 );
```

Watch out - only single backticks (')!

Or:

```
INSERT INTO EMPLOYEE(Fname, Lname, Ssn) VALUES (
   'Hugo', 'Chavez', '324323');
```

DELETING data

```
DELETE FROM EMPLOYEE
    WHERE Lname = 'Chavez';

DELETE FROM EMPLOYEE
    WHERE Fname = 'Hugo';

DELETE FROM EMPLOYEE;
```

The last example clears the entire table - usually no questions asked!

UPDATING data

```
UPDATE EMPLOYEE
SET Fname = 'Hugo', Lname = 'Chavez'
    WHERE Ssn = '324323345';

UPDATE EMPLOYEE
SET Fname = 'Hugo', Lname = 'Chavez';
```

Everybody becomes Hugo Chavez - again, no questions asked.

A word of warning

DBMSs are **expert tools** - they don't usually ask for confirmation, and there typically is no "undo" button.

Think before you execute a SQL statement, or make backups (e.g., database dump).

Many DBMSs return the number of **affected rows** for DML statements.

- E.g., "17 rows deleted", "32453627 rows updated", …
- Allows sanity check whether command was correct

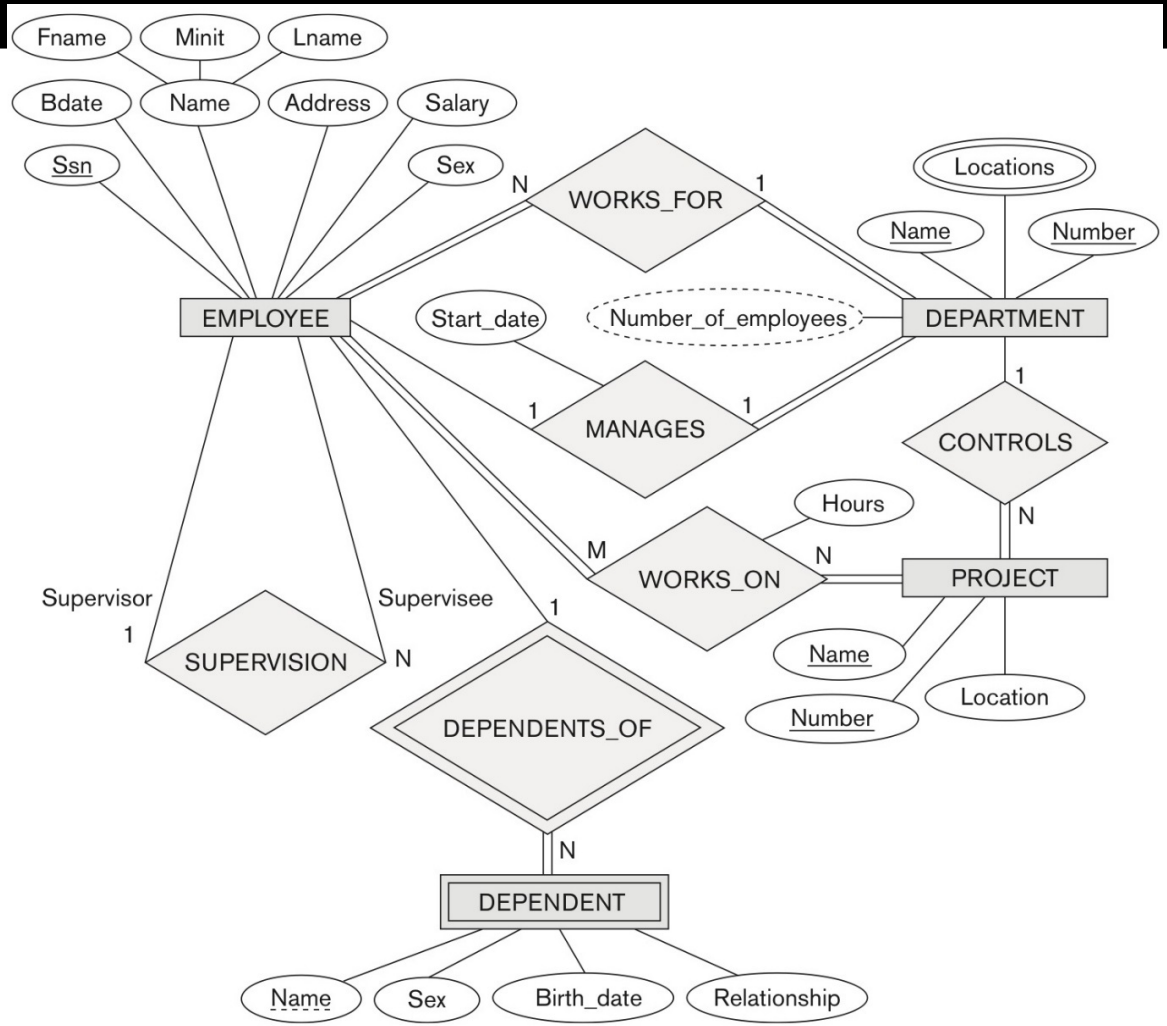
How do get to tables from E(E)R

How do we get from our **logical domain model** in E(E)R to database tables?

Mapping (mostly) straight-forward:

- Relations become tables
- Attributes become table attributes
- Attribute types and constraints need to be introduced

Let's discuss a few special cases.



Copyright (c) 2011 Pearson Education

Mapping Relationships

1:N Relationship

Straight-forward to map via PK/FK (FK needs to be in the ’N’ table)

1:1 Relationship

Map either like a 1:N relationship, *or*

Merge entities into single table (has all attributes from both relations)

N:M Relationships need to be resolved via a cross-reference table

E.g., many EMPLOYEEs WORKS_ON many PROJECTs

```
CREATE TABLE EMPLOYEE(…);
CREATE TABLE PROJECT(…);
CREATE TABLE EMP_PROJ(
    Employee CHAR(9), Project CHAR(9),
    PRIMARY KEY(Employee, Project),
    FOREIGN KEY(Employee) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY(Project) REFERENCES PROJECT(Number)
);
```

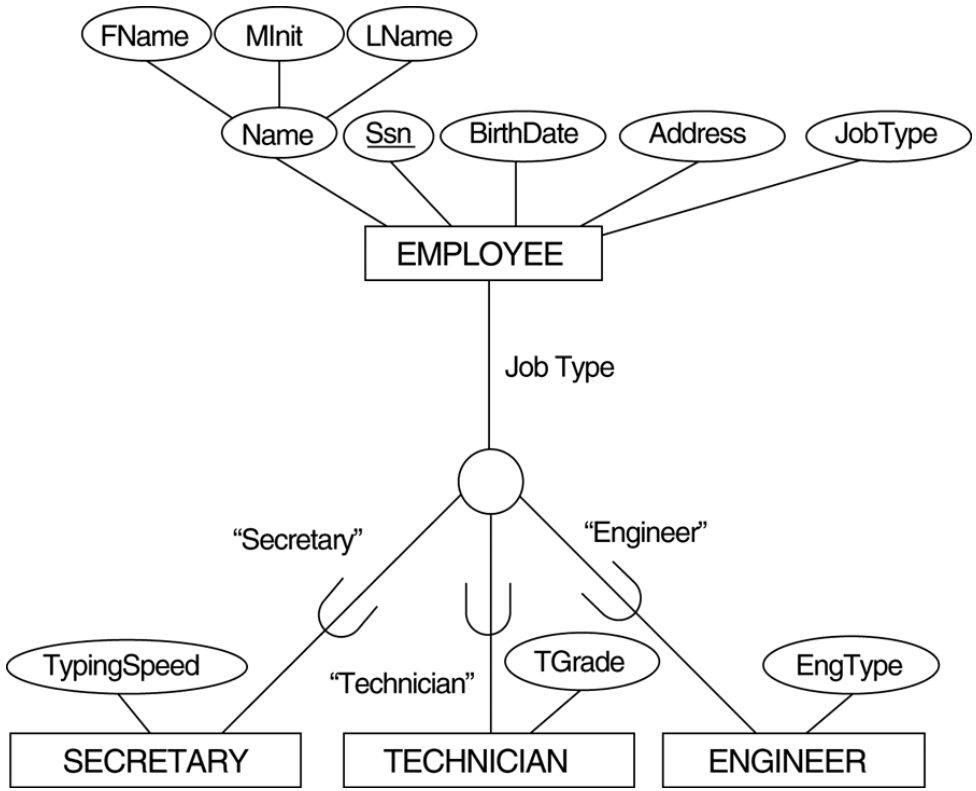
Same basic approach can also be used for:

Mapping relationships with an **relationship attribute**

Mapping **composed attributes** (attribute becomes a separate table)

Mapping **multi-value attributes** (attribute again becomes a separate table)

Mapping Inheritance



Four different approaches, depending on type of inheritance.

Remember, EER inheritance comes in four flavors:

Disjoint / total
Disjoint / partial
Overlapping / total
Overlapping / partial

Correct mapping to tables depends on the flavor.

Disjoint / Partial

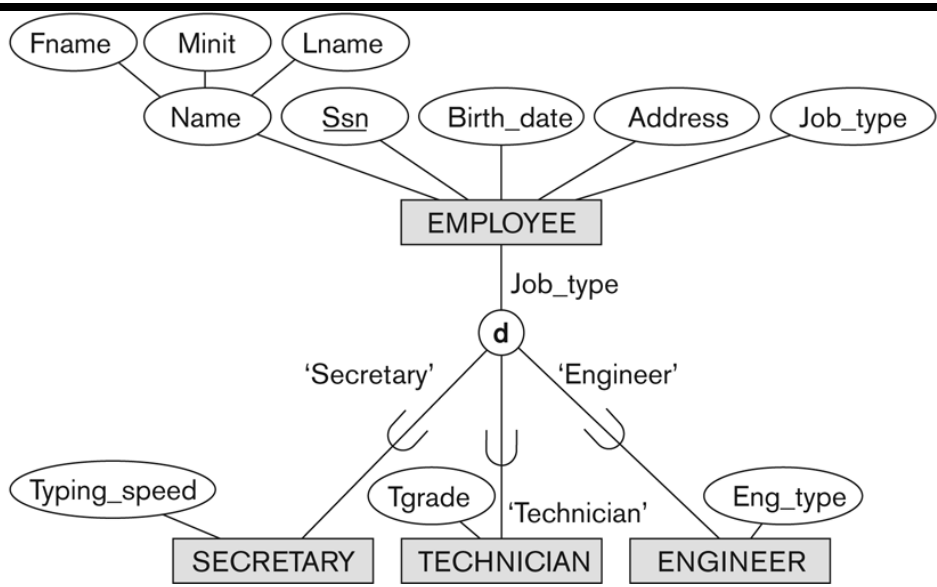


Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.

Map as four tables, one for the super-entity and one each for the sub-entities.

Use FKs to handle relationships.

(a) EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType

SECRETARY

SSN	TypingSpeed

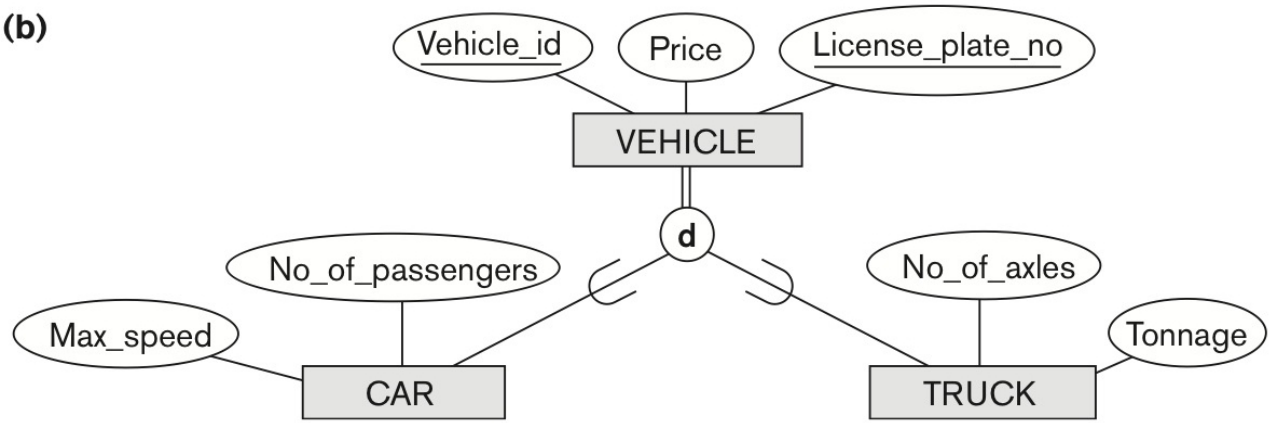
TECHNICIAN

SSN	TGrade

ENGINEER

SSN	EngType

Disjoint / Total



Map as two tables:

(b)

CAR

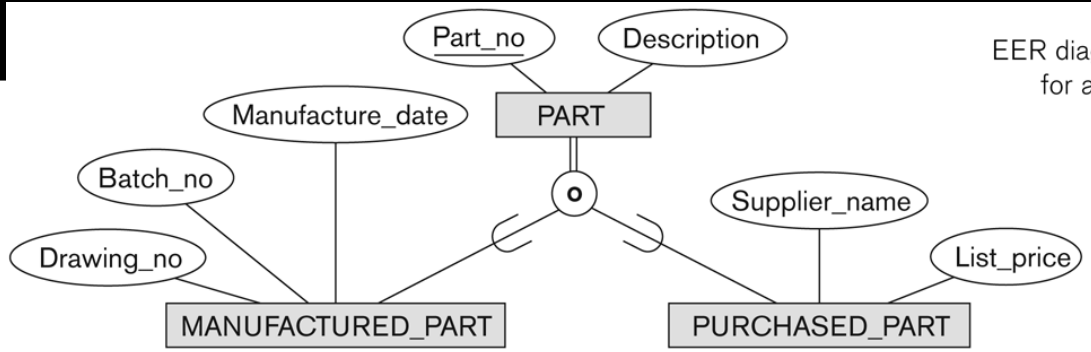
VehicleId	LicensePlateNo	Price	MaxSpeed	NoOfPassengers

TRUCK

VehicleId	LicensePlateNo	Price	NoOfAxles	

Alternatively, map as three tables (VEHICLE, CAR, TRUCK) and join via FKs, just like for disjoint / partial.

Overlapping / Total



Map as a single table with all attributes:

(d) PART

PartNo	Description	MFlag	DrawingNo	ManufactureDate	BatchNo	PFlag	SupplierName	ListPrice

All of the attributes coming from sub-entities need to be allowed to be NULL.

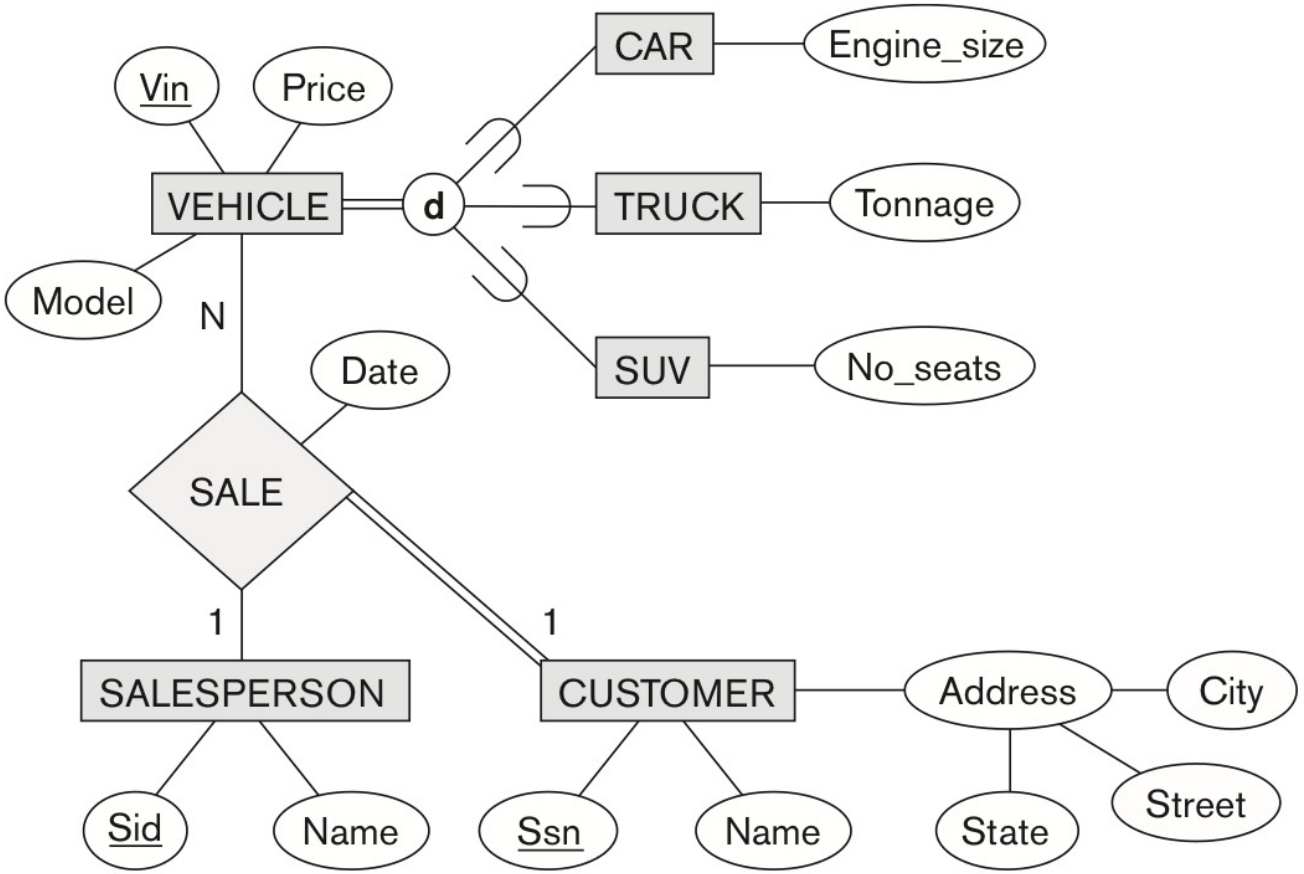
You don’t necessarily need the sub-entity flags.

Overlapping / Partial

Map the same as Overlapping / Total

Main difference is that for some rows, all attributes coming from sub-entities will be NULL.

Another Exercise -
Create the table model for this



Key Takeaways

Basic DDL and DML SQL commands

You need to train using them as part of the assignments!

Strategies for mapping E(E)R to SQL tables

It’s a fairly mechanical process, but it is important that you understand the general concepts