DDL & DML

⏪ Recap

- Subquery
- Aggregation
- Grouping
- Joining

> I know she's around 5'5" (65") or 5'7" (67"). She has red hair and she drives a Tesla Model S.

```sql
select id
from drivers_license
where hair_color = "red"
and car_make = "Tesla"
and car_model = "Model S"
and height between 65 and 67
```

> I know that she attended the SQL Symphony Concert 3 times in December 2017.

```sql
select person_id
from facebook_event_checkin
where event_name like "%SQL Symphony Concert%"
and date between 20170101 and 20171231
group by person_id
having count(*)=3
```

```sql
-- Subquery
select *
from person
where license_id in (...)
and id in (...)
```

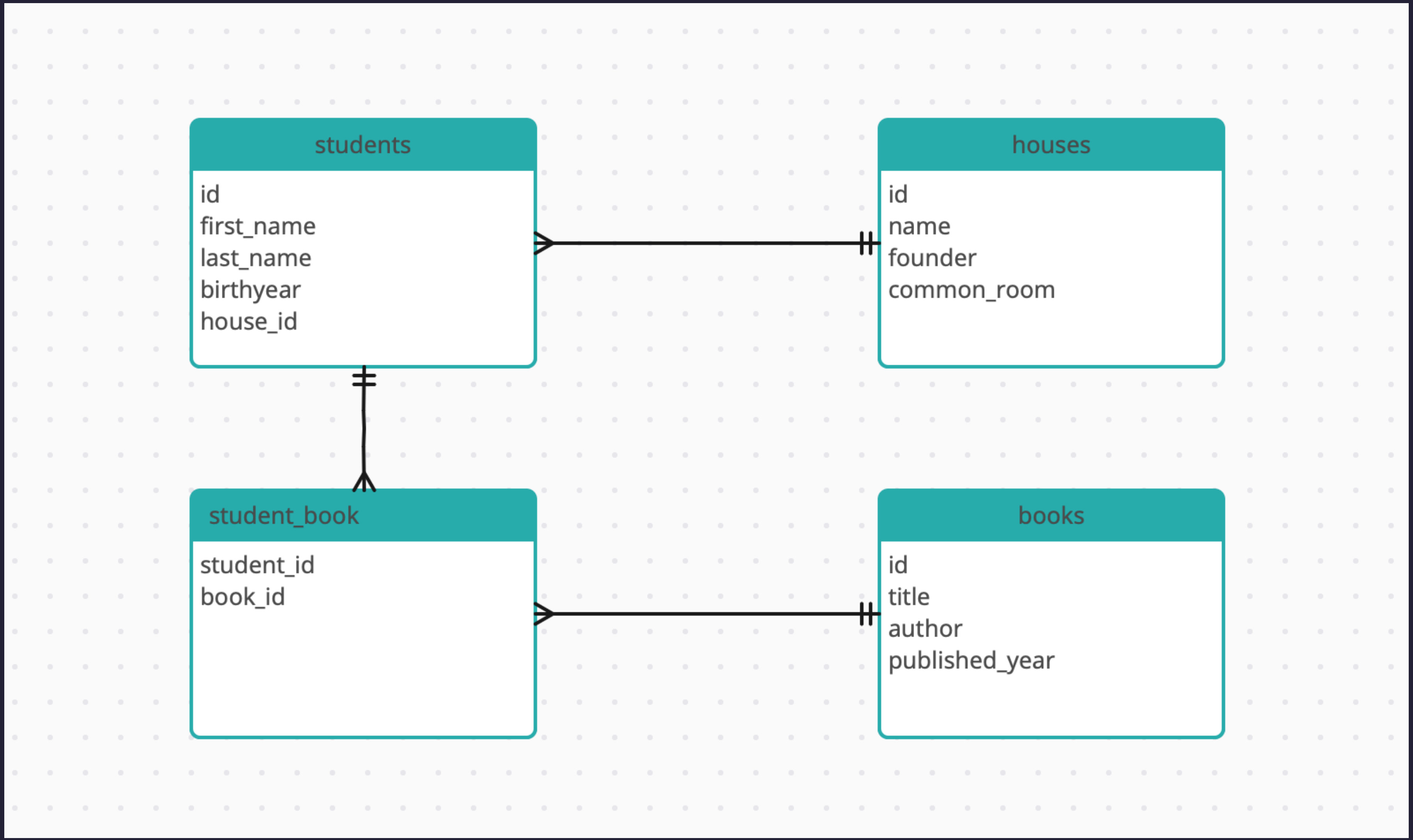
DDL: Data Definition Language

- `CREATE TABLE`: **create a new table**
- `ALTER TABLE`: modify a table
- `DROP TABLE`: remove a table

DML: Data Manipulation Language

- `INSERT INTO`: **insert data into a table**
- `UPDATE`: update data in a table
- `DELETE FROM`: remove data from a table

Expanding the Harry Potter database



students

id
first_name
last_name
birthyear
house_id

houses

id
name
founder
common_room

student_book

student_id
book_id

books

id
title
author
published_year

`CREATE TABLE`: Create a new table

```sql
CREATE TABLE books (
    -- columns and data types
    id numeric,
    title text,
    author text,
    published_year numeric,
    -- constraints
    PRIMARY KEY (id),
    CHECK (published_year >= 1900)
);
```

Data types

`numeric`

- `int`
- `float`

…

`text`

- `varchar(n)`
- `text`

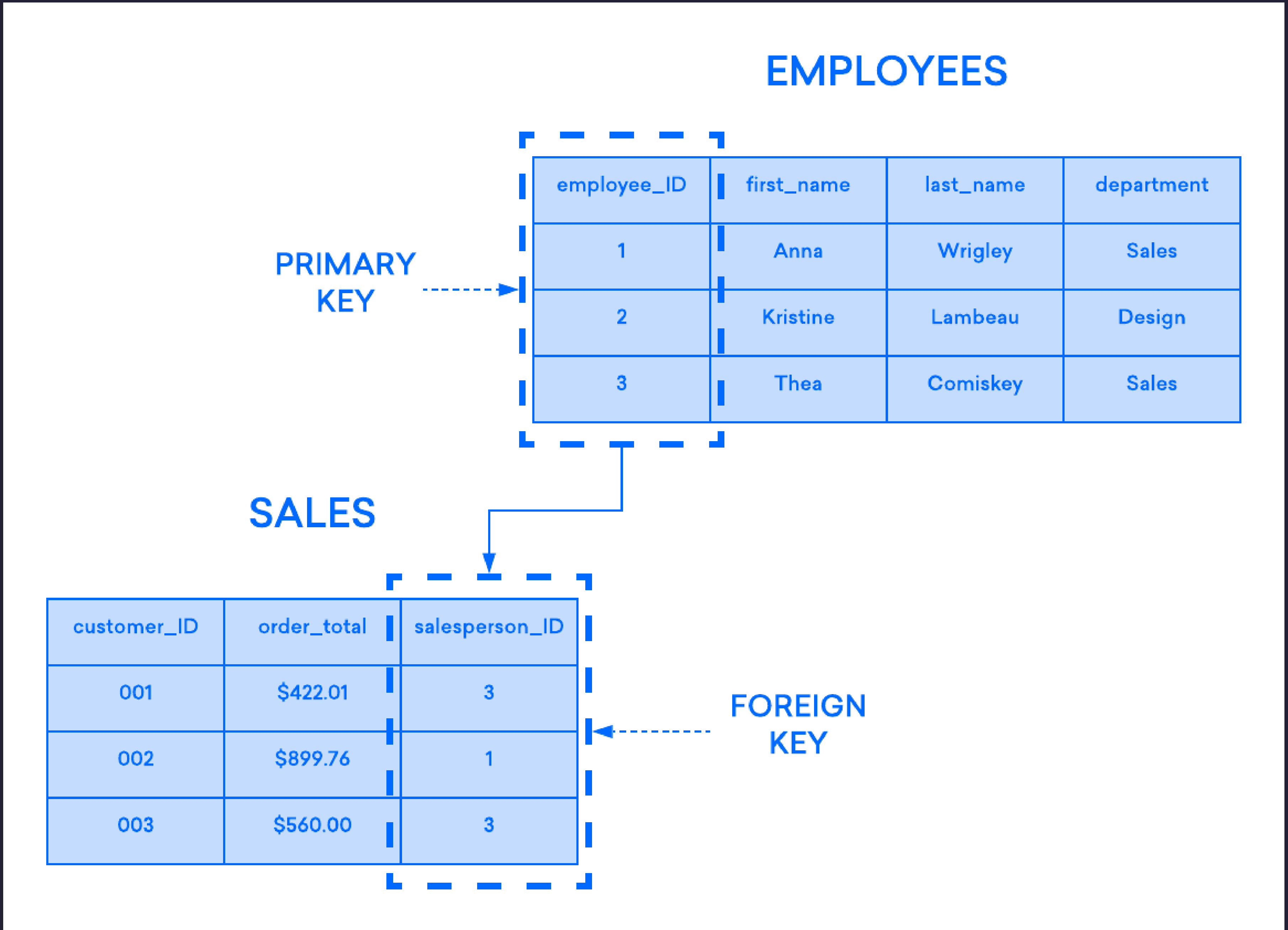
…

https://www.sqlite.org/datatype3.html

Constraints to limit what values can be inserted into columns

- `primary key` : `unique` and `not null`
- `foreign key` : references another table
- `not null` : must have a value
- `unique` : must be unique
- `check(exp)` : must satisfy the expression

https://www.sqlite.org/lang_createtable.html



EMPLOYEES

employee_ID	first_name	last_name	department
1	Anna	Wrigley	Sales
2	Kristine	Lambeau	Design
3	Thea	Comiskey	Sales

PRIMARY KEY

SALES

customer_ID	order_total	salesperson_ID
001	$422.01	3
002	$899.76	1
003	$560.00	3

FOREIGN KEY

`INSERT INTO`: Insert data into a table

```sql
INSERT INTO books (id, title)
VALUES (1, 'Harry Potter and the Sorcerer''s Stone');

INSERT INTO books (id, title, author, published_year)
VALUES (1, 'Harry Potter and the Sorcerer''s Stone', 'J.K. Rowling', 1997);

INSERT INTO books
VALUES (1, 'Harry Potter and the Sorcerer''s Stone', 'J.K. Rowling', 1997);

INSERT INTO books (id, title, author, published_year)
VALUES
    (1, 'Harry Potter and the Sorcerer''s Stone', 'J.K. Rowling', 1997),
    (2, ...),
    (3, ...);
```

Create books and student_book

```
CREATE TABLE books (
    id int,
    title text,
    author text,
    published_year int,
    -- primary key: id
    PRIMARY KEY (id)
);
CREATE TABLE student_book (
    student_id int,
    book_id int,
    -- composite PK: student_id and book_id together form a PK
    PRIMARY KEY (student_id, book_id),
    -- FK: student_id to refreence the students table
    FOREIGN KEY (student_id) REFERENCES students(id),
    -- FK: book_id to reference the books table
    FOREIGN KEY (book_id) REFERENCES books(id)
);
```

Insert data (with column names)

```sql
INSERT INTO books (id, title, author, published_year) VALUES
(201, 'Harry Potter and the Philosopher''s Stone', 'J.K. Rowling', 1997),
(202, 'Harry Potter and the Chamber of Secrets', 'J.K. Rowling', 1998),
(203, 'Harry Potter and the Prisoner of Azkaban', 'J.K. Rowling', 1999),
(204, 'Harry Potter and the Goblet of Fire', 'J.K. Rowling', 2000),
(205, 'Harry Potter and the Order of the Phoenix', 'J.K. Rowling', 2003),
(206, 'Harry Potter and the Half-Blood Prince', 'J.K. Rowling', 2005),
(207, 'Harry Potter and the Deathly Hallows', 'J.K. Rowling', 2007);
```

Insert data (without column names)

```sql
INSERT INTO student_book VALUES
(101, 201), (101, 202), (101, 203), (102, 201), (102, 203),
(102, 204), (103, 201), (103, 204), (104, 202), (104, 205),
(105, 203), (105, 206), (106, 201), (106, 203), (106, 207),
(107, 205), (107, 206), (108, 204), (108, 206), (109, 203),
(109, 207), (110, 205), (111, 204), (112, 201), (113, 203),
(114, 202), (115, 205), (116, 204), (117, 201), (118, 202);
```

💻 Add Movies table to Harry Potter database

- table name: `movies`
- columns:
  - `id` int
  - `title` text
  - `director` text
  - `year` int
- primary key: `id`

🖥️ Insert data into Movies table

```
1, 'Harry Potter and the Sorcerer''s Stone', 'Chris Columbus', 2001
2, 'Harry Potter and the Chamber of Secrets', 'Chris Columbus', 2002
3, 'Harry Potter and the Prisoner of Azkaban', 'Alfonso Cuarón', 2004
4, 'Harry Potter and the Goblet of Fire', 'Mike Newell', 2005
5, 'Harry Potter and the Order of the Phoenix', 'David Yates', 2007
6, 'Harry Potter and the Half-Blood Prince', 'David Yates', 2009
7, 'Harry Potter and the Deathly Hallows – Part 1', 'David Yates', 2010
8, 'Harry Potter and the Deathly Hallows – Part 2', 'David Yates', 2011
```

DDL: `ALTER TABLE`

- add columns

```
ALTER TABLE mytable
ADD column_name DataType OptionalTableConstraint

ALTER TABLE mytable
ADD column_name DataType OptionalTableConstraint
DEFAULT OptionalDefaultValue;
```

- remove columns

```
ALTER TABLE mytable
DROP column_name;
```

- rename the table

```
ALTER TABLE mytable
RENAME TO new_table_name;
```

DDL: `DROP TABLE`

```sql
DROP TABLE mytable;
DROP TABLE IF EXISTS mytable;
```

DML: UPDATE table

```
UPDATE books
SET published_year = 1998
WHERE id = 1;
```

DML: `DELETE FROM` table

```sql
DELETE FROM books
WHERE id = 1;
```

🖥️ Update and delete data from the Movies table

1. remove director column from the movies table
2. add a new column `director_id` to the movies table
3. delete the movie with id 1 from the movies table
4. update the year of the movie with id 2 to 2003