WEEK 2

RDBMS ARCHITECTURE

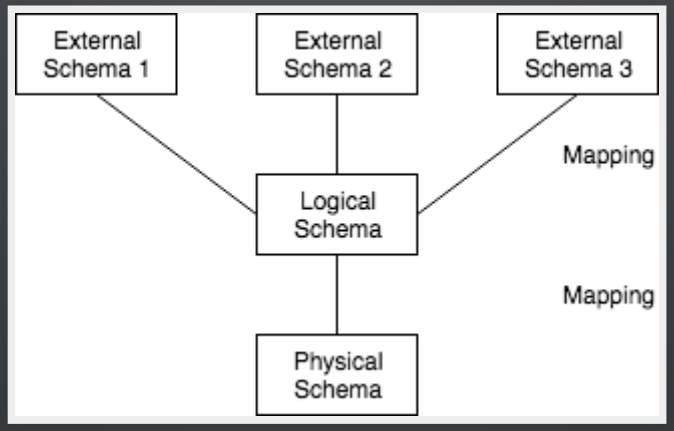
NORMALIZATION

MORE SQL

DATABASE ARCHITECTURE

3 SCHEMA ARCHITECTURE

- External / User Views
- Conceptual / Logical
- Internal / Physical



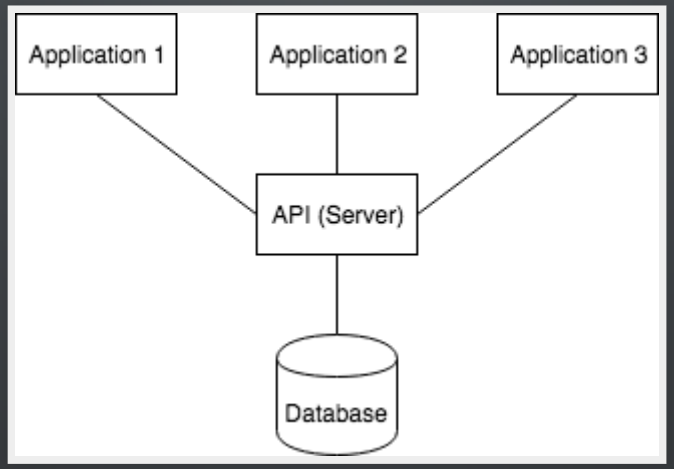
SCHEMA LANGUAGES

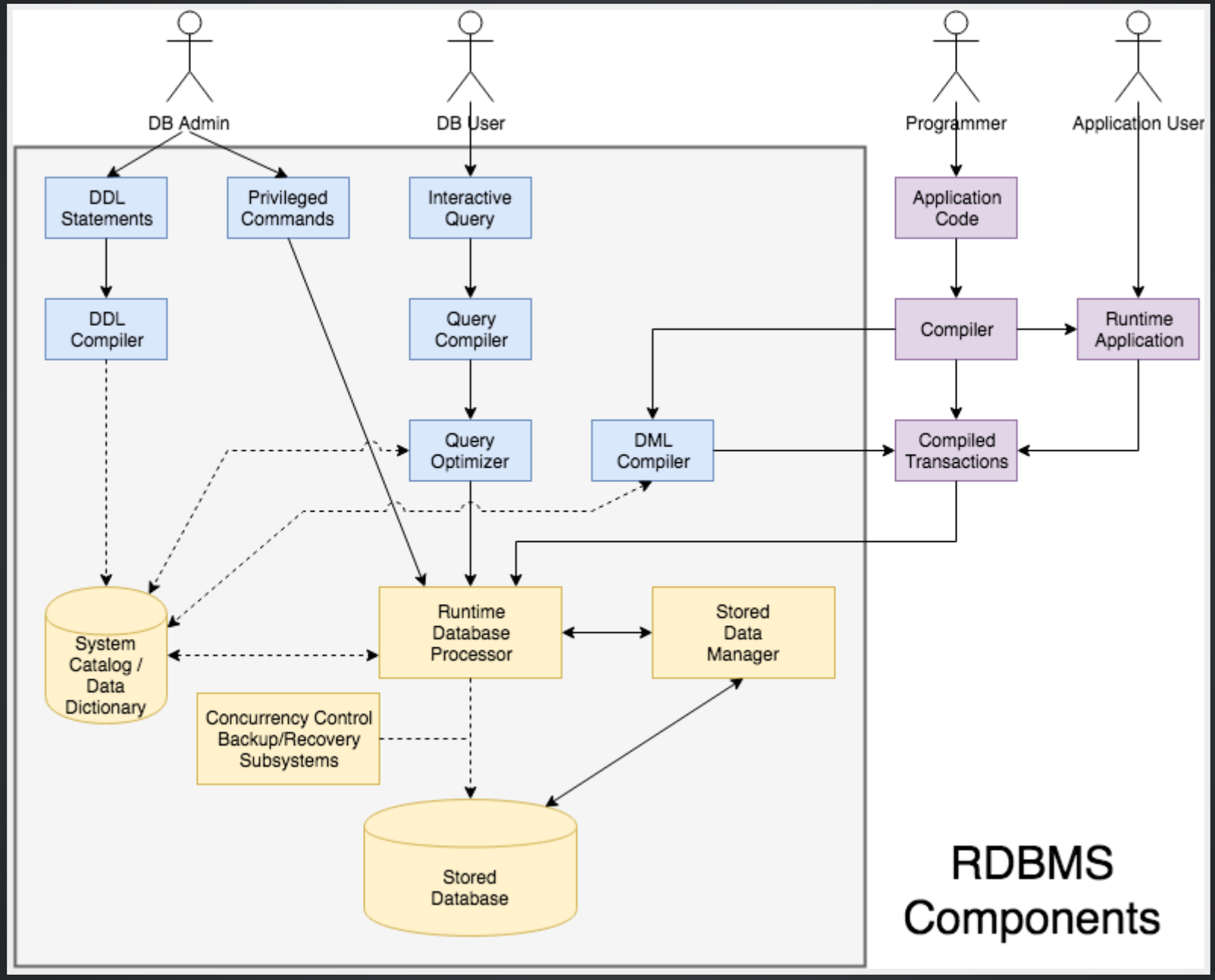
- VDL: view definition
- DML: data manipulation
- DDL: data definition
- SDL: storage definition

- All rolled into SQL
  - Except SDL

3 TIER ARCHITECTURE

- Client Application(s)
- API (Server)
- Database




DB Admin
DB User
Programmer
Application User
DDL Statements
Privileged Commands
Interactive Query
Application Code
DDL Compiler
Query Compiler
Compiler
Runtime Application
Query Optimizer
DML Compiler
Compiled Transactions
System Catalog / Data Dictionary
Runtime Database Processor
Stored Data Manager
Concurrency Control Backup/Recovery Subsystems
Stored Database
RDBMS Components

RDBMS UTILITIES

- Loading
- Backup
- Monitoring
- Reorganization
  - Cleans up mess from deleted rows

SCHEMA NORMALIZATION

RULES OF THUMB

1. Avoid Ambiguous Attributes
2. Avoid Redundant Data
3. Avoid NULLs
4. Avoid Spurious JOINs

1. AVOID AMBIGUOUS ATTRIBUTES

- Semantics of the attributes should be clear
- Design tables that are easy to explain
- Don't mix attributes from multiple entity types in a single relation

OBJECTS

ObjectID	Title	Type	DateCreated	ArtistID	ArtistName	ArtistDOB
1	Irises	Painting	1889	1	Van Gogh, Vincent	1853-03-30
2	Spring	Painting	1881	2	Manet, Édouard	1832-01-23
3	Olympia	Painting	1862	2	Manet, Édouard	1832-01-23

EXHIBITION_LOANS

ObjID	ExName	ExStart	ExEnd	LoanStart	LoanEnd	Museum	Mgr	Address	Phone
1	Impress Me	2018-01-01	2018-06-01	2017-12-01	2018-07-07	The Getty	PersonX	LA, CA, USA	(310) 440-7300
2	Impress Me	2018-01-01	2018-06-01	2017-11-09	2018-08-01	The Getty	PersonY	LA CA, USA	(310) 440-7300
3	Impress Me	2018-01-01	2018-06-01	2017-10-11	2018-06-22	Musée d'Orsay	PersonZ	Paris, France	+33 (0)1 40 49 48 14

2. AVOID REDUNDANT DATA

- Repeated values wastes storage space
- Also leads to anomalies
  - insertion anomalies
  - deletion anomalies
  - modification anomalies

INSERTION ANOMALIES

OBJECTS

ObjectID	Title	Type	DateCreated	ArtistID	ArtistName	ArtistDOB
1	Irises	Painting	1889	1	Van Gogh, Vincent	1853-03-30
2	Spring	Painting	1881	2	Manet, Édouard	1832-01-23
3	Olympia	Painting	1862	2	Manet, Édouard	1832-01-23

- Inserting an object requires artist data (or NULLs)
  - Prone to inconsistency
- How do you insert an artist w/o an object?
  - Use NULLs
  - After an object is inserted w/ that artist?

DELETION ANOMALIES

OBJECTS

ObjectID	Title	Type	DateCreated	ArtistID	ArtistName	ArtistDOB
1	Irises	Painting	1889	1	Van Gogh, Vincent	1853-03-30
2	Spring	Painting	1881	2	Manet, Édouard	1832-01-23
3	Olympia	Painting	1862	2	Manet, Édouard	1832-01-23

- What if you delete the last object referring to a particular artist?
  - Artist is lost from the database

MODIFICATION ANOMALIES

OBJECTS

ObjectID	Title	Type	DateCreated	ArtistID	ArtistName	ArtistDOB
1	Irises	Painting	1889	1	Van Gogh, Vincent	1853-03-30
2	Spring	Painting	1881	2	Manet, Édouard	1832-01-23
3	Olympia	Painting	1862	2	Manet, Édouard	1832-01-23

- What if we update the Manager name for a museum?
  - We must update all of the rows with that museum
  - Leads to inconsistency

3. AVOID NULLS

- NULLs also waste space
- Can lead to ambiguity when performing:
  - JOINs
  - aggregate operations (count, sum, max)
  - comparisons (WHERE x > y)
- NULLs can have multiple meanings:
  - N/A, unknown, missing

4. AVOID SPURIOUS JOINS

- Design tables that can be joined on Primary Key-Foreign Key pairs
- Joining on non-prime attributes leads to spurious tuples

OBJECTS

Title	**Type**	**DateCreated**	**ArtistName**	**ArtistDOB**
Irises	Painting	1889	Van Gogh, Vincent	1853-03-30
Spring	Painting	1881	Manet, Édouard	1832-01-23
Olympia	Painting	1862	Manet, Édouard	1832-01-23
Spring	Sculpture	2001	Dude, Some	NULL

EXHIBITION_LOANS

Title	**ExName**	**ExStart**	**ExEnd**	**LoanStart**	**LoanEnd**	**Museum**	**Mgr**	**Address**
Irises	Impress Me	2018-01-01	2018-06-01	2017-12-01	2018-07-07	The Getty	PersonX	LA, CA, USA
Spring	Impress Me	2018-01-01	2018-06-01	2017-11-09	2018-08-01	The Getty	PersonX	LA CA, USA
Olympia	Impress	2018-01-01	2018-06-01	2017-10-11	2018-06-22	Musée d'Orsay	PersonZ	Paris, France

SPURIOUS JOIN

```
SELECT o.Title, o.ArtistName, e.Museum
FROM Objects o, Exhibition_Loans e
WHERE e.ExName="Impress Me"
AND e.Title = o.Title;
```

Title	ArtistName	Museum
Irises	Van Gogh, Vincent	The Getty
Spring	Manet, Édouard	The Getty
Olympia	Manet, Édouard	Musée d'Orsay
Spring	Dude, Some	The Getty

NORMALIZATION

NORMALIZATION BASICS

- A formal method of analysis to minimize redundancy and anomalies
- Has multiples forms (levels), higher is better
- Based on primary keys and functional dependencies
- Normalizing tables individually is not sufficient
  - Taken together, the tables must also ensure nonadditive joins (no spurious tuples)

FUNCTIONAL DEPENDENCY (FD)

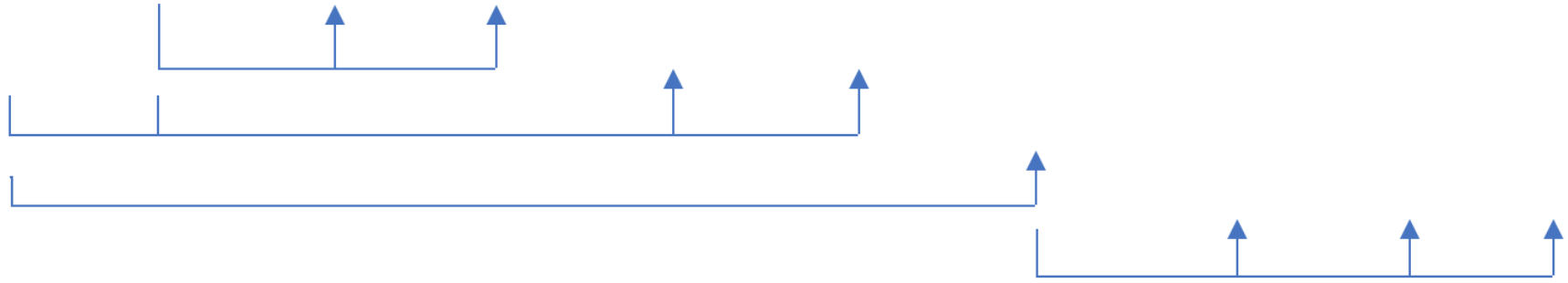
- A constraint between 2 sets of attributes
- Denoted by X -> Y
  - X and Y can each have more than one attribute
  - Means that the values of Y are determined by the values of X

> *For any 2 tuples in the database that have the same set of X values, they must also have the same set of Y values.*

FUNCTIONAL DEPENDENCIES

EXHIBITION_LOANS

ObjID	ExName	ExStart	ExEnd	LoanStart	LoanEnd	Museum	Mgr	Address	Phone
1	Impress Me	2018-01-01	2018-06-01	2017-12-01	2018-07-07	The Getty	PersonX	LA, CA, USA	(310) 440-7300
2	Impress Me	2018-01-01	2018-06-01	2017-11-09	2018-08-01	The Getty	PersonX	LA CA, USA	(310) 440-7300
3	Impress Me	2018-01-01	2018-06-01	2017-10-11	2018-06-22	Musée d'Orsay	PersonZ	Paris, France	+33 (0)1 40 49 48 14



FIRST NORMAL FORM (1NF)

Values of any attribute must be atomic and singular

MOVIES

Title	Year	Genres
The Shape of Water	2017	Fantasy, Romance
Get Out	2017	Horror, Mystery, Thriller

TECHNIQUES FOR 1NF

1. Move the violating attribute to another table
    - e.g. Movies, Genres, HasGenre
    - Decomposition
2. Repeat the row for each value in the attribute
    - Creates redundancy
3. Add attributes for each possible value
    - e.g. Genre1, Genre2, Genre3, etc.
    - Must know maximum number of values
    - Creates NULLs

2ND NORMAL FORM (2NF)

Every nonprime attribute in a table is fully functionally dependent (FFD) on the primary key

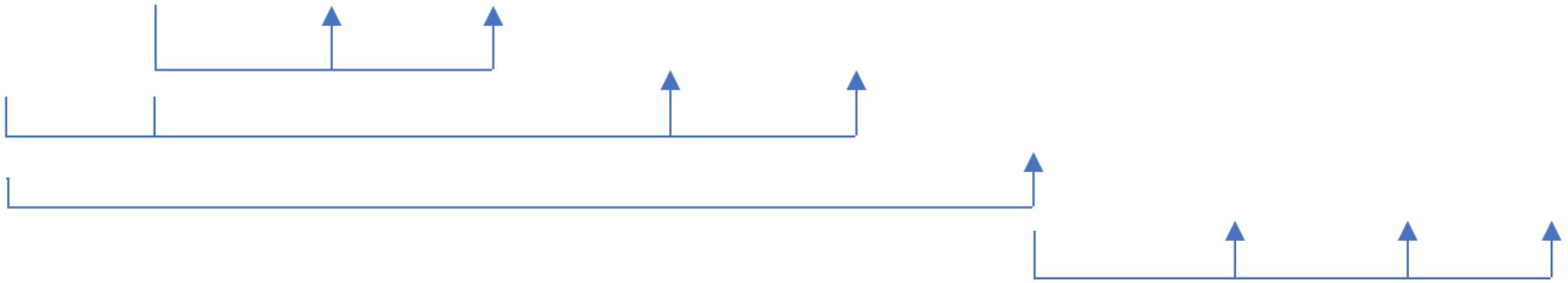
- Nonprime = not part of the primary key (PK)
- FFD = removal of any part of the PK erodes the FD
- Only applies to tables with composite PKs

TECHNIQUE FOR 2NF

- Decomposition: new table for each partial key

EXHIBITION_LOANS

ObjID	ExName	ExStart	ExEnd	LoanStart	LoanEnd	Museum	Mgr	Address	Phone
1	Impress Me	2018-01-01	2018-06-01	2017-12-01	2018-07-07	The Getty	PersonX	LA, CA, USA	(310) 440-7300
2	Impress Me	2018-01-01	2018-06-01	2017-11-09	2018-08-01	The Getty	PersonX	LA CA, USA	(310) 440-7300
3	Impress Me	2018-01-01	2018-06-01	2017-10-11	2018-06-22	Musée d'Orsay	PersonZ	Paris, France	+33 (0)1 40 49 48 14



3RD NORMAL FORM (3NF)

No nonprime attribute is transitively dependent on the primary key

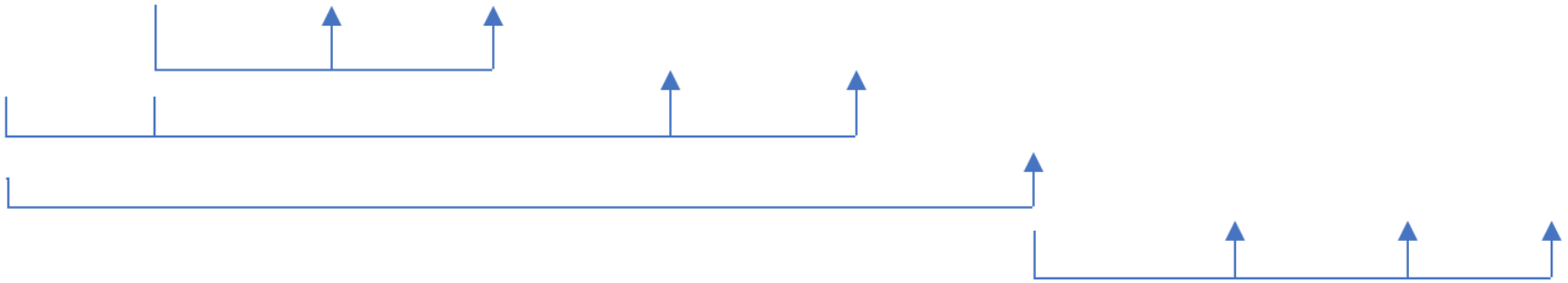
- i.e. All attributes should be FFD on the PK, not on another attribute

TECHNIQUE FOR 3NF

- Decompose: new table for transitive dependencies

EXHIBITION_LOANS

ObjID	ExName	ExStart	ExEnd	LoanStart	LoanEnd	Museum	Mgr	Address	Phone
1	Impress Me	2018-01-01	2018-06-01	2017-12-01	2018-07-07	The Getty	PersonX	LA, CA, USA	(310) 440-7300
2	Impress Me	2018-01-01	2018-06-01	2017-11-09	2018-08-01	The Getty	PersonX	LA CA, USA	(310) 440-7300
3	Impress Me	2018-01-01	2018-06-01	2017-10-11	2018-06-22	Musée d'Orsay	PersonZ	Paris, France	+33 (0)1 40 49 48 14



BOYCE-CODD NORMAL FORM (BCNF)

Essentially the same as 3NF, with small addition:

> *A nonprime attribute cannot functionally determine a prime attribute*

- Any table that is BCNF, is also 3NF
- Most tables in 3NF are also BCNF

4TH NORMAL FORM (4NF)

A table cannot have a multivalued dependency (MVD)

- MVD: occurs when 2 independent 1:N relationships exist in the same table
  - Results in repeated values
  - Often occurs in all-key tables

TECHNIQUE FOR 4NF

- Decompose each dependency into a new table

Band	Album	Musician
Queens of the Stone Age	Like Clockwork	Josh Homme
Queens of the Stone Age	Songs for the Deaf	Josh Homme
Queens of the Stone Age	Songs for the Deaf	Troy Van Leeuwen
Queens of the Stone Age	Songs for the Deaf	Dave Grohl
Nirvana	Nevermind	Dave Grohl
Nirvana	Nevermind	Kurt Cobain
Nirvana	Nevermind	Chris Novaselic

5TH NORMAL FORM?

- Dont' bother
  - For join depedencies among 3 or more tables
- In practice, most DBAs aim for 3NF, which is critical
- BCNF, 4NF, and 5NF are subtle and rarely needed

MORE SQL

ON DELETE

What should happen if a foreign key is deleted?

```
CREATE TABLE chapter(
    number  INTEGER,
    title   VARCHAR(256),
    bookID  INTEGER REFERENCES book ON DELETE CASCADE
);
```

Options include: `SET DEFAULT`, `SET NULL`, `CASCADE`, `RESTRICT`, `NO ACTION`

ON UPDATE

What happens if the foreign key is changed?

e.g. FK is an ISBN, changed from 10 to 13 digits

```
CREATE TABLE chapter(
    number  INTEGER,
    title   VARCHAR(256),
    isbn    VARCHAR(15) REFERENCES book.isbn
            ON DELETE CASCADE
            ON UPDATE CASCADE
);
```

Same options as `ON DELETE`

ALTER TABLE

Want to rename a table?

```
ALTER TABLE actors RENAME TO thesbians;
```

What if you want to add or remove a column?

```
ALTER TABLE chapter ADD COLUMN pageCount INTEGER;
```

```
ALTER TABLE chapter DROP COLUMN pageCount;
```

`DROP COLUMN` not supported by some DBs

MORE ABOUT SELECT

- DISTNICT
- LIKE
- LIMIT & OFFSET

DISTINCT

- Imagine a DB with 3 tables:
  - musician: `id`, `name`
  - album: `id`, `title`, `year`
  - played_on: `musicianID`, `albumID`, `role`
- How do you get all the roles a musician has ever performed without repeats?

```
SELECT DISTINCT musician.name, played_on.role
FROM musician, played_on
WHERE played_on.musicianID=musician.id;
```

LIKE

- Fuzzy matching for text with regular expressions
- Wildcard Characters
  - `%` represents zero, one, or more characters
  - `_` represents a single character
- e.g. Find books with "information" in the title

```
SELECT * FROM book WHERE title LIKE '%information%';
```

LIMIT

Too many results?

`LIMIT` them!

```
SELECT * FROM books WHERE title LIKE '%the%' LIMIT 20;
```

OFFSET

What if you want to page through the results?

Use `LIMIT` and `OFFSET` together

```
SELECT * FROM books WHERE title LIKE '%the%' LIMIT 20 OFFSET 20;
```

NESTED QUERIES

- Sometimes you need to perform one query and use the results in another
- You can do that in one statement with nested queries

```
SELECT title
FROM album, band
WHERE album.bandid=band.id
AND band.name="Metallica"
AND album.id IN (
    SELECT id
    FROM album
    ORDER BY earnings DESC
    LIMIT 100
    );
```

AGGREGATE FUNCTIONS

- Standard SQL has aggregate functions that can be called within a query.
- Each RDBMS also supports it's own functions
- SQLite3 supports these functions:
  - `min(X)`,`max(X)`,`avg(X)`,`sum(X)`,`total(X)`
  - `count(*)`,`count(X)`
  - `group_concat(X, Y)`

```
SELECT count(title) FROM films;
```

```
SELECT max(earnings), avg(earnings) FROM films;
```

OUTER JOINS

- What happens when you make a query using a FK that can have NULLs?
  - If you use a `NATURAL INNER JOIN` (aka equijoin), the rows with NULLs will not be included

```
SELECT bookt.title, author.name
FROM book, author
WHERE book.authorid=author.id;
```

- If you want to include rows with NULLs you have to do an `OUTER JOIN`

OUTER JOINS

- To keep all the elements from the first table use a `LEFT OUTER JOIN`

```
SELECT book.title, author.name
FROM book
LEFT OUTER JOIN ON book.authorid=author.id;
```

Title	Name
Ancillary Justice	Ann Leckie
Arabian Nights	
Night Circus	Erin Morgenstern

OUTER JOINS

- To keep all the elements from the second table use a `RIGHT OUTER JOIN`

```
SELECT book.title, author.name
FROM book
RIGHT OUTER JOIN ON book.authorid=author.id;
```

Title	Name
Ancillary Justice	Ann Leckie
	Joshua Gomez
Night Circus	Erin Morgenstern

OUTER JOINS

- To keep all the elements from both tables use a `FULL OUTER JOIN`

```
SELECT book.title, author.name
FROM book
FULL OUTER JOIN ON book.authorid=author.id;
```

Title	Name
Ancillary Justice	Ann Leckie
Arabian Nights	
Night Circus	Erin Morgenstern
	Joshua Gomez

GROUP BY

- You can arrange your results in groups
- This works well with aggregate functions

```
SELECT count(id), genre
FROM albums
GROUP BY genre
ORDER BY count(id) DESC;
```

HAVING

- What if you want to add a condition based on the aggregate function?
- Use the `HAVING` clause

```
SELECT count(id), genre
FROM albums
GROUP BY genre
HAVING count(id) > 1000
ORDER BY count(id) DESC;
```

EOL

On to the exercise!

After a break