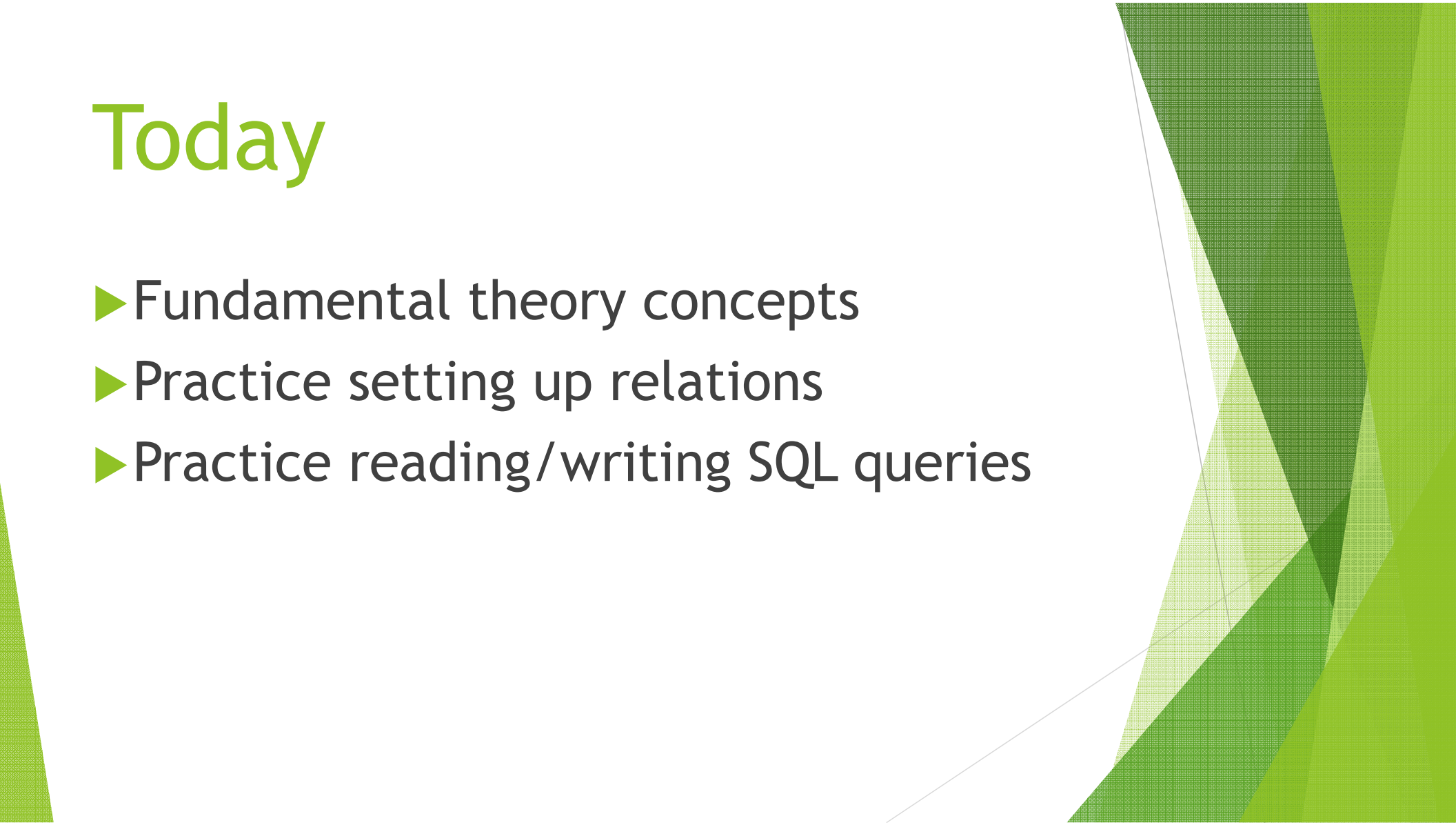
COMP 353 - Databases

This slides are prepared by:

- Nicole Parmentier
  - nicole.parmentier@mail.concordia.ca

Today

- Fundamental theory concepts
- Practice setting up relations
- Practice reading/writing SQL queries



What is a database?

- Collection of data stored long-term, organized in a way that is meaningful and logical

- e.g. contact list, employee files, etc.

What is a database?

- DB: Database
- DBS: Database System
- DMBS: Database Management System

DBS = DB + DBMS
system = data + management

DMBS

- Offers convenient and secure access
- Transaction Management:
  - Atomicity (all or nothing)
  - Consistency (follows rules)
  - Isolation (transactions do not affect each other)
  - Durability (data survives failures)

DMBS

- Better than a simple file directory:

Minimize data redundancy to avoid inconsistency

Relational Data Model

- A data model describes data and the relationships among that data
- Relational model organizes data (attributes) into named tables (relations)
- e.g.

  Hero = (name, gameTitle, class, level, hitPoints)

  Game = (title, platform, releaseYear, genre)

Define a relation

- Define a relation for a book:
- Book = (title, publisher, releaseDate, subject, price, pageCount, inPublicDomain)
- RelationName = (string, string, date, string, float, integer, boolean)
- Many different relations can be defined according to the application requirements

Levels of Data

- View (external): how data is seen by users
- Conceptual (logical) Schema: defined relations, an outcome of database design
- Physical (internal) Schema: storage and index structures associated with relations

Note: The levels are independent of the ones below them

Structured Query Language (SQL)

- Data Definition Language (DDL)
  - Syntax for declaring tables, indexes, views, constraints, etc.
- Data Manipulation Language (DML)
  - Queries
  - Transactions (data modification)
  - Schema creation/modification

SQL Queries

- An SQL query follows the form:

```
SELECT . . .
FROM . . .
WHERE . . . ;
```

-**select** which attributes are output

-which relations to draw data **from**

-**where** that data meets these conditions (boolean)

SQL Queries

- Relation schema:

  Game = (title, platform, releaseYear, genre)

- Query: Find the titles of all games for the platform ‘N64’ which are stored in the database
- Query in SQL:

```
SELECT title
FROM Game
WHERE platform = ‘N64’;
```

SQL Queries

- WHERE clause is a boolean, can be as complex as you saw in COMP 232:
- Book = (title, publisher, releaseDate, subject, price, pageCount, inPublicDomain)

```
SELECT *
FROM Book
WHERE (price < 19.99 AND (pageCount < 42 OR pageCount >= 420)) OR title = 'Brave New World';
```

- What is this query?

SQL Queries

- FROM clause can use multiple tables (relations), coupled through a Cartesian product

Hero = (name, gameTitle, class, level, hitPoints)

Game = (title, platform, releaseYear, genre)

```
SELECT Hero.name, Game.title , Game.genre
FROM Game, Hero
WHERE Hero.gameTitle = Game.title
        AND Game.releaseYear > 1980;
```

SQL Queries

- Relation schemas:

Hero = (name, gameTitle, class, level, hitPoints)

Game = (title, platform, releaseYear, genre)

- Instances of the relations:

name	gameTitle	class	level	hitPoints
Mario	Super Mario 64	Plumber	0	1
Link	Legend of Zelda	Warrior	1	3

title	platform	releaseYear	genre
Super Mario 64	N64	1996	Platformer
Pikmin	Gamecube	2001	Puzzle-strategy

Cartesian Product

name	gameTitle	class	level	hitPoints
Mario	Super Mario 64	Plumber	0	1
Link	Legend of Zelda	Warrior	1	3

title	platform	releaseYear	genre
Super Mario 64	N64	1996	Platformer
Pikmin	Gamecube	2001	Puzzle

Hero x Game

name	gameTitle	class	level	hitPoints	title	platform	releaseYear	genre
Mario	Super Mario 64	Plumber	0	1	Super Mario 64	N64	1996	Platformer
Mario	Super Mario 64	Plumber	0	1	Pikmin	Gamecube	2001	Puzzle
Link	Legend of Zelda	Warrior	1	3	Super Mario 64	N64	1996	Platformer
Link	Legend of Zelda	Warrior	1	3	Pikmin	Gamecube	2001	Puzzle

```
SELECT Hero.name, Game.title , Game.genre
FROM Game, Hero
WHERE Hero.gameTitle = Game.title
        AND Game.releaseYear > 1980;
```

name	gameTitle	class	level	hitPoints	title	platform	releaseYear	genre
Mario	Super Mario 64	Plumber	0	1	Super Mario 64	N64	1996	Platformer
Mario	Super Mario 64	Plumber	0	1	Pikmin	Gamecube	2001	Puzzle
Link	Legend of Zelda	Warrior	1	3	Super Mario 64	N64	1996	Platformer
Link	Legend of Zelda	Warrior	1	3	Pikmin	Gamecube	2001	Puzzle

- Query Result:

name	title	genre
Mario	Super Mario 64	Platformer

Aggregation in SQL

- Operations done in the SELECT clause to summarize the data
  - SUM, AVG, MIN, MAX, COUNT
- Will often want to use GROUP BY and HAVING to filter our data

```
SELECT publisher, SUM(price), COUNT(title)
FROM Book
GROUP BY publisher
HAVING MIN(price) > 1.00;
```

Aggregation in SQL

- HAVING applies a condition on the groups themselves (after grouping), as opposed to a condition in the WHERE clause which applies to the individual entries/tuples/rows

```
SELECT publisher, SUM(price), COUNT(title)
FROM Book
WHERE price > 1.00
GROUP BY publisher;
```

Queries in SQL

- Time to test your knowledge on a problem which uses everything we've covered today (and more!)

name	gameTitle	class	level	hitPoints
Mario	Super Mario 64	Plumber	4	1
Link	Legend of Zelda: Wind Waker	Warrior	1	3
Olimar	Pikmin 2	Thief	2	100
Kirby	Kirby 64: The Crystal Shards	Shapeshifter	5	50
Leon	Resident Evil 4	Gunner	3	200
Luigi	Luigi's Mansion	Ghostbuster	0	5
Louie	Pikmin 2	Assistant	0	100

```
SELECT Game.genre,
       COUNT(Hero.name)
FROM Game, Hero
WHERE Hero.level >= 1
   AND Hero.gameTitle =
   Game.title
GROUP BY Game.genre
HAVING MIN(Game.releaseYear) > 1999
ORDER BY Game.genre DESC;
```

title	platform	releaseYear	genre
Super Mario 64	N64	1996	Platformer
Pikmin 2	Gamecube	2004	Puzzle
Legend of Zelda: Wind Waker	Gamecube	2003	Adventure
Kirby 64: The Crystal Shards	N64	2000	Platformer
Luigi's Mansion	Gamecube	2001	Adventure
Resident Evil 4	Gamecube	2005	Adventure

name	gameTitle	class	level	hitPoints	platform	release Year	genre
Mario	Super Mario 64	Plumber	4	1	N64	1996	Platformer
Link	Legend of Zelda: Wind Waker	Warrior	1	3	Gamecube	2003	Adventure
Olimar	Pikmin 2	Thief	2	100	Gamecube	2004	Puzzle
Louie	Pikmin 2	Assistant	0	100	Gamecube	2004	Puzzle
Kirby	Kirby 64: The Crystal Shards	Shapeshifter	5	50	N64	2000	Platformer
Leon	Resident Evil 4	Gunner	3	200	Gamecube	2005	Adventure
Luigi	Luigi's Mansion	Ghostbuster	0	5	Gamecube	2001	Adventure

```
SELECT Game.genre,
       COUNT(Hero.name)
FROM Game, Hero
WHERE Hero.level >= 1
   AND Hero.gameTitle =
   Game.title
GROUP BY Game.genre
HAVING MIN(Game.releaseYear) > 1999
ORDER BY Game.genre DESC;
```

*Note: here we do the Cartesian product and apply the title filter from the WHERE clause immediately in order to keep the table manageable. Observe that Olimar and Louie both matched with the Game entry for Pikmin 2

name	gameTitle	class	level	hitPoints	platform	release Year	genre
Mario	Super Mario 64	Plumber	4	1	N64	1996	Platformer
Link	Legend of Zelda: Wind Waker	Warrior	1	3	Gamecube	2003	Adventure
Olimar	Pikmin 2	Thief	2	100	Gamecube	2004	Puzzle
~~Louie~~	~~Pikmin 2~~	~~Assistant~~	~~0~~	~~100~~	~~Gamecube~~	~~2004~~	~~Puzzle~~
Kirby	Kirby 64: The Crystal Shards	Shapeshifter	5	50	N64	2000	Platformer
Leon	Resident Evil 4	Gunner	3	200	Gamecube	2005	Adventure
~~Luigi~~	~~Luigi's Mansion~~	~~Ghostbuster~~	~~0~~	~~5~~	~~Gamecube~~	~~2001~~	~~Adventure~~

```
SELECT Game.genre,
       COUNT(Hero.name)
FROM Game, Hero
WHERE Hero.level >= 1
   AND Hero.gameTitle =
   Game.title
GROUP BY Game.genre
HAVING MIN(Game.releaseYear) > 1999
ORDER BY Game.genre DESC;
```

name	gameTitle	class	level	hitPoints	platform	release Year	genre
Mario	Super Mario 64	Plumber	4	1	N64	1996	Platformer
Kirby	Kirby 64: The Crystal Shards	Shapeshifter	5	50	N64	2000	Platformer
Olimar	Pikmin 2	Thief	2	100	Gamecube	2004	Puzzle
Leon	Resident Evil 4	Gunner	3	200	Gamecube	2005	Adventure
Link	Legend of Zelda: Wind Waker	Warrior	1	3	Gamecube	2003	Adventure

```
SELECT Game.genre,
       COUNT(Hero.name)
FROM Game, Hero
WHERE Hero.level >= 1
  AND Hero.gameTitle =
  Game.title
GROUP BY Game.genre
HAVING MIN(Game.releaseYear) > 1999
ORDER BY Game.genre DESC;
```

*Note: here I have reordered the entries by genre as the beginning of the GROUP BY action, but it is only finished once the entries are collapsed into single rows by genre and once the desired attributes are aggregated. I have extended this process into several steps here so it is easier to see what's happening with the data in the table and the various filters being applied

name	gameTitle	class	level	hitPoints	platform	release Year	genre
~~Mario~~	~~Super Mario 64~~	~~Plumber~~	~~4~~	~~1~~	~~N64~~	~~1996~~	~~Platformer~~
~~Kirby~~	~~Kirby 64: The Crystal Shards~~	~~Shapeshifter~~	~~5~~	~~50~~	~~N64~~	~~2000~~	~~Platformer~~
Olimar	Pikmin 2	Thief	2	100	Gamecube	2004	Puzzle
Leon	Resident Evil 4	Gunner	3	200	Gamecube	2005	Adventure
Link	Legend of Zelda: Wind Waker	Warrior	1	3	Gamecube	2003	Adventure

```
SELECT Game.genre,
       COUNT(Hero.name)
FROM Game, Hero
WHERE Hero.level >= 1
   AND Hero.gameTitle =
   Game.title
GROUP BY Game.genre
HAVING MIN(Game.releaseYear) > 1999
ORDER BY Game.genre DESC;
```

*Note: Mario and Kirby are grouped together in the Platformer genre, and this group's minimum release year is Mario's 1996. Because this group does not meet the HAVING condition, the whole group (including Kirby) is filtered out

genre	COUNT(Hero.name)
Adventure	2
Puzzle	1

```
SELECT Game.genre,
       COUNT(Hero.name)
FROM Game, Hero
WHERE Hero.level >= 1
   AND Hero.gameTitle =
   Game.title
GROUP BY Game.genre
HAVING MIN(Game.releaseYear) > 1999
ORDER BY Game.genre DESC;
```

genre	COUNT(Hero.name)
Puzzle	1
Adventure	2

```
SELECT Game.genre,
       COUNT(Hero.name)
FROM Game, Hero
WHERE Hero.level >= 1
   AND Hero.gameTitle =
   Game.title
GROUP BY Game.genre
HAVING MIN(Game.releaseYear) > 1999
ORDER BY Game.genre DESC;
```

*Note: ORDER BY - DESC sorts the values in decreasing/descending order. For a string, this means the opposite of alphabetical order, like in java. We also have ASC for ascending order, which is the default of ORDER BY.

These slides were made referencing the course slides from Prof Shiri, as well as the textbook:
A First Course in Database Systems by Ullman and Widom, third edition