SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20031
Computing Technology Design Project

Week 08
Queries and Catering for Concurrency

Database Development Lifecycle

1. Planning
2. Requirement gathering
3. Conceptual design
4. Logical design
5. Physical design
6. **Construction**
7. **Implementation** & rollout
8. Ongoing support

Outline

1. Mapping requirements to queries
2. SQL query (SELECT)
3. Transactions
4. (Enhanced) SQL view
5. Project update

(A) Mapping requirement to query

Requirement => Query

Requirement:

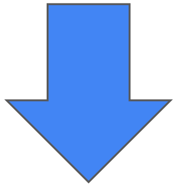
Find all countries in Southeast Asia that have more than 70 millions people. Sort the result by name, in ASC.

```
SELECT * from Country
    WHERE Region = ‘Southeast Asia’
      AND Population > 70000000
    ORDER BY Name
```

Requirement => Query (complex ones)

Requirement:

Find all countries with life-expectancies and populations higher than their averages in the same region. Sort the result by region, life-expectancy and population (ASC).



?

(B) SQL Query

Reference: Chapters 01, 08

SQL Query

```
SELECT * from Country
    WHERE Name like ‘%v%’
    ORDER BY Name
    LIMIT 10, 5;
```

Database: world.sql

- To select data values from the database
  - satisfying requirements of certain use case/user story
- Data values:
  - all or some selected columns
- Sources:
  - one table, or
  - several (joined) tables

SQL Query: selected columns

```
SELECT Name AS Country, Code AS ISO

    FROM Country

    WHERE Name like ‘%v%’

    ORDER BY Code
```

- Keyword ‘AS’: to name output columns using an alias
  - helps avoid naming conflict or name a computation result
- Question:
  - what does the above query do?

Counting rows

```
SELECT Count(*) AS Count
     FROM Country
     WHERE Name like '%v%'
```

≠

```
SELECT Count(lifeexpectancy)
AS CountLife
     FROM Country
     WHERE Name like '%v%'
```

- Find number of rows satisfying a condition
- Find number of **non-null** values of a column
- Question:
  - use a query to check the difference (17) in outputs of the two queries above?

SQL Join

```
SELECT a.artist AS Artist, a.title AS Album, t.track_number
AS 'Track Num',
    t.title AS Track, t.duration AS Seconds
  FROM album AS a
  JOIN track AS t ON a.id = t.album_id
  ORDER BY a.artist, a.title, t.track_number;
```

Database: album.sql

- To get data from multiple related tables
- A join is defined based on FK-PK pair of two tables
- Multiple join clauses can be specified
- Question:
  - What does the example query produce?
  - Translate the query into natural language (NL)?

Aggregates

Reference: Chapter 08

```
SELECT Region, COUNT(*) AS Count
  FROM Country
  GROUP BY Region
  ORDER BY Count DESC, Region;
```

Database: world.sql

- Aggregate (computed) data obtained from a query
- Typically used with GROUP BY to aggregate based on common values
- Question:
  - What does the example query produce?
  - Translate the query into natural language (NL)?

Aggregates (2)

```
SELECT a.title AS Album, COUNT(t.track_number) as Tracks
  FROM track AS t
  JOIN album AS a
    ON a.id = t.album_id
  GROUP BY a.id
  HAVING Tracks >= 10
  ORDER BY Tracks DESC, Album
```

Database: `album.sql`

- **HAVING** clause to filter groups of rows satisfy GROUP BY
  - **esp.** used for **aggregate columns**

Aggregates (3)

```
SELECT a.title AS Album, COUNT(t.track_number) as Tracks
  FROM track AS t
  JOIN album AS a
    ON a.id = t.album_id
  WHERE a.artist = 'The Beatles'
  GROUP BY a.id
  HAVING Tracks >= 10
  ORDER BY Tracks DESC, Album
```

- **WHERE** clause to filter based on columns in SELECT list
    - ***cannot** be* used with aggregate columns

Aggregate functions (1)

```
SELECT COUNT(*) FROM Country;
SELECT COUNT(Population) FROM Country;
SELECT COUNT(DISTINCT HeadOfState) FROM Country;
SELECT AVG(Population) FROM Country;
SELECT Region, AVG(Population)
  FROM Country GROUP BY Region;
SELECT Region, MIN(Population), MAX(Population)
  FROM Country GROUP BY Region;
SELECT Region, SUM(Population)
  FROM Country GROUP BY Region;
```

- **COUNT**: 3 variations
- **AVG**, **MIN**, **MAX**, **SUM**
- All can be used with a column with Group By

(C) Transactions

Reference: Chapter 09

Transaction

- **Transaction**: a group of data operations that are handled as one unit
  - succeed all of fail
- Ensure consistent state of database
- **Concurrent** operations: e.g. caused by multiple users
  - can be grouped into transactions
- **Performance**: many writes can be grouped to perform together

Consistency (example: test db)

```
START TRANSACTION;
  INSERT INTO widgetSales(inv_id, quan, price )
    VALUES (1, 5, 500);
  UPDATE widgetInventory
    SET onhand = (onhand - 5)
    WHERE id = 1;
COMMIT;
```

- Transaction (start):
  - Insert into one table
  - Update data in a related table
  - Commit

Performance

```
BEGIN TRANSACTION;
  -- performs this 100 times
  INSERT INTO widgetSales(inv_id, quan, price )
   VALUES (1, 5, 500);
END TRANSACTION;
```

SQLite syntax

- Observe the difference in execution times: 100 INSERTs
  - use **SQLiteStudio** to see the total execution time
- Observation 1 (without transaction):
- Observation 2 (with transaction):
  - ~20 times faster !

(D) SQL View

Reference: Chapter 11

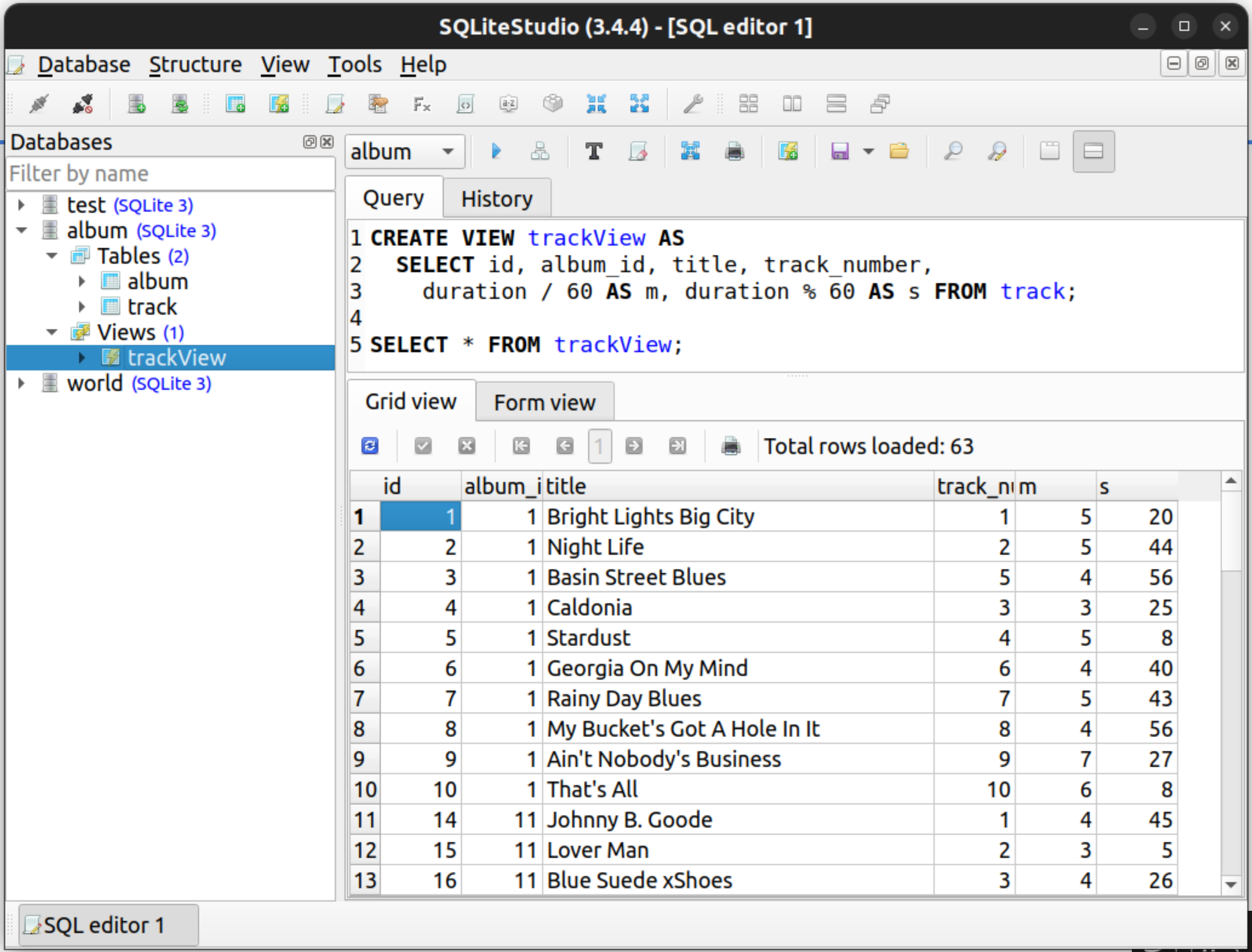
SQL view

```
CREATE VIEW trackView AS
  SELECT id, album_id, title, track_number,
    duration / 60 AS m, duration % 60 AS s FROM track;

SELECT * FROM trackView;
```

- View is a saved query for reuse
  - esp. useful if query is complex
- Can be used as a table

Example



```
CREATE VIEW trackView AS
  SELECT id, album_id, title, track_number,
    duration / 60 AS m, duration % 60 AS s FROM track;

SELECT * FROM trackView;
```

Total rows loaded: 63

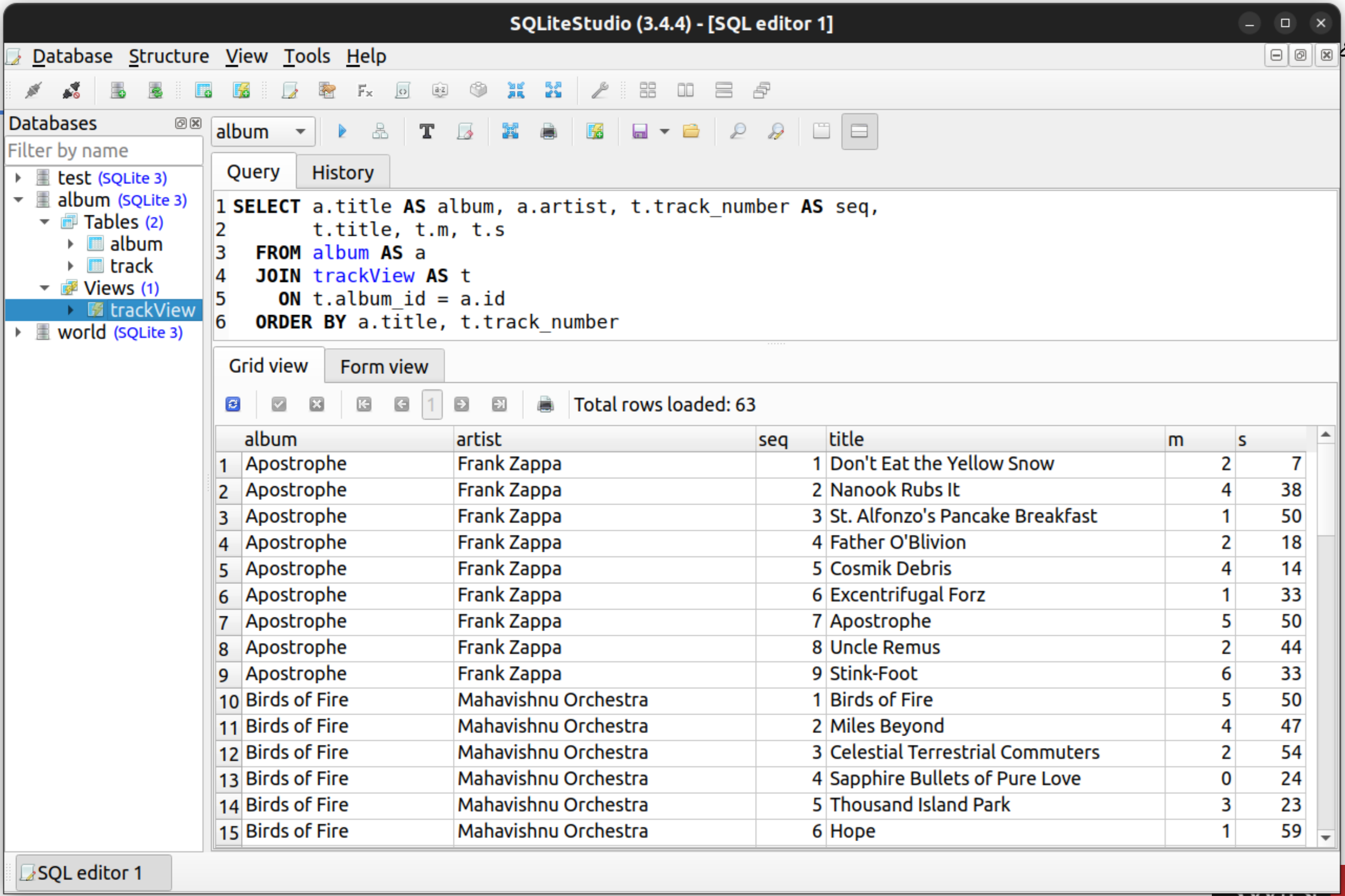
	id	album_i	title	track_nı	m	s
1	1	1	Bright Lights Big City	1	5	20
2	2	1	Night Life	2	5	44
3	3	1	Basin Street Blues	5	4	56
4	4	1	Caldonia	3	3	25
5	5	1	Stardust	4	5	8
6	6	1	Georgia On My Mind	6	4	40
7	7	1	Rainy Day Blues	7	5	43
8	8	1	My Bucket's Got A Hole In It	8	4	56
9	9	1	Ain't Nobody's Business	9	7	27
10	10	1	That's All	10	6	8
11	14	11	Johnny B. Goode	1	4	45
12	15	11	Lover Man	2	3	5
13	16	11	Blue Suede xShoes	3	4	26

SQL view (2)

```
SELECT a.title AS album, a.artist, t.track_number AS seq,
t.title, t.m, t.s
  FROM album AS a
  JOIN trackView AS t
    ON t.album_id = a.id
  ORDER BY a.title, t.track_number
```

- Using view as a table

Example



```
SELECT a.title AS album, a.artist, t.track_number AS seq,
       t.title, t.m, t.s
  FROM album AS a
  JOIN trackView AS t
    ON t.album_id = a.id
  ORDER BY a.title, t.track_number
```

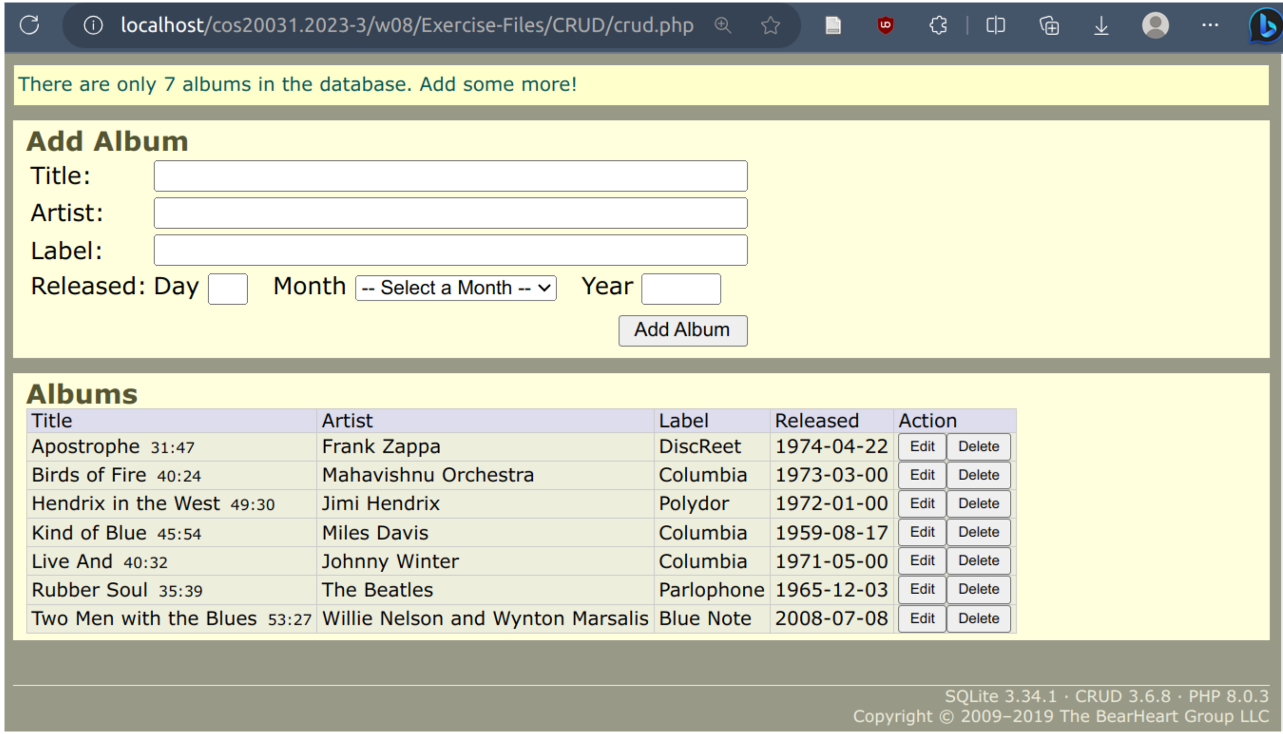
Total rows loaded: 63

	album	artist	seq	title	m	s
1	Apostrophe	Frank Zappa	1	Don't Eat the Yellow Snow	2	7
2	Apostrophe	Frank Zappa	2	Nanook Rubs It	4	38
3	Apostrophe	Frank Zappa	3	St. Alfonzo's Pancake Breakfast	1	50
4	Apostrophe	Frank Zappa	4	Father O'Blivion	2	18
5	Apostrophe	Frank Zappa	5	Cosmik Debris	4	14
6	Apostrophe	Frank Zappa	6	Excentrifugal Forz	1	33
7	Apostrophe	Frank Zappa	7	Apostrophe	5	50
8	Apostrophe	Frank Zappa	8	Uncle Remus	2	44
9	Apostrophe	Frank Zappa	9	Stink-Foot	6	33
10	Birds of Fire	Mahavishnu Orchestra	1	Birds of Fire	5	50
11	Birds of Fire	Mahavishnu Orchestra	2	Miles Beyond	4	47
12	Birds of Fire	Mahavishnu Orchestra	3	Celestial Terrestrial Commuters	2	54
13	Birds of Fire	Mahavishnu Orchestra	4	Sapphire Bullets of Pure Love	0	24
14	Birds of Fire	Mahavishnu Orchestra	5	Thousand Island Park	3	23
15	Birds of Fire	Mahavishnu Orchestra	6	Hope	1	59

(D) CRUD applications

CRUD application (1)

- URL: Embedding SQL
- Code set: in the Exercise File of the course
- Language: PHP
- DBMS: SQLite
- Deploy and Demo: on XAMPP



There are only 7 albums in the database. Add some more!

Add Album

Title:

Artist:

Label:

Released: Day Month -- Select a Month -- Year

Add Album

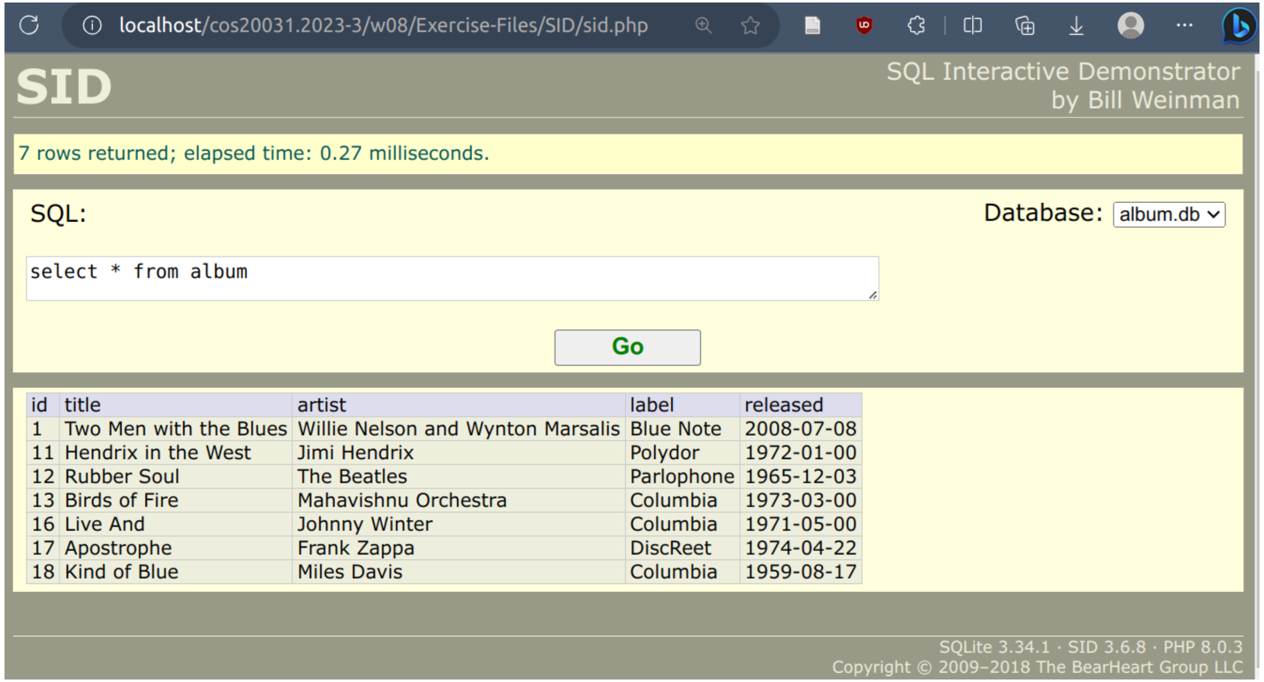
Albums

Title	Artist	Label	Released	Action
Apostrophe 31:47	Frank Zappa	DiscReet	1974-04-22	Edit Delete
Birds of Fire 40:24	Mahavishnu Orchestra	Columbia	1973-03-00	Edit Delete
Hendrix in the West 49:30	Jimi Hendrix	Polydor	1972-01-00	Edit Delete
Kind of Blue 45:54	Miles Davis	Columbia	1959-08-17	Edit Delete
Live And 40:32	Johnny Winter	Columbia	1971-05-00	Edit Delete
Rubber Soul 35:39	The Beatles	Parlophone	1965-12-03	Edit Delete
Two Men with the Blues 53:27	Willie Nelson and Wynton Marsalis	Blue Note	2008-07-08	Edit Delete

SQLite 3.34.1 · CRUD 3.6.8 · PHP 8.0.3
Copyright © 2009–2019 The BearHeart Group LLC

SQL Application: SID

- URL: [Embedding SQL]
- Code set: in the Exercise File of the course
- Language: PHP
- DBMS: SQLite
- Deploy and Demo: on [XAMPP]


localhost/cos20031.2023-3/w08/Exercise-Files/SID/sid.php
SID
SQL Interactive Demonstrator
by Bill Weinman

7 rows returned; elapsed time: 0.27 milliseconds.

SQL:

Database: album.db

```
select * from album
```

id	title	artist	label	released
1	Two Men with the Blues	Willie Nelson and Wynton Marsalis	Blue Note	2008-07-08
11	Hendrix in the West	Jimi Hendrix	Polydor	1972-01-00
12	Rubber Soul	The Beatles	Parlophone	1965-12-03
13	Birds of Fire	Mahavishnu Orchestra	Columbia	1973-03-00
16	Live And	Johnny Winter	Columbia	1971-05-00
17	Apostrophe	Frank Zappa	DiscReet	1974-04-22
18	Kind of Blue	Miles Davis	Columbia	1959-08-17

SQLite 3.34.1 · SID 3.6.8 · PHP 8.0.3
Copyright © 2009–2018 The BearHeart Group LLC

(E) Project update

Project update

- Map user stories to queries
- Test queries on your database
  - use aggregates, view, transactions
- Update database design (if need to)
  - Remember: iterative development (agile)
- Plan a CRUD application
- Update project documentation (Confluence)

Tutorial & Workshop

See Canvas.