COMP9120

Week 12: Review

Semester 2, 2022

Professor Athman Bouguettaya
School of Computer Science

Acknowledgement of Country

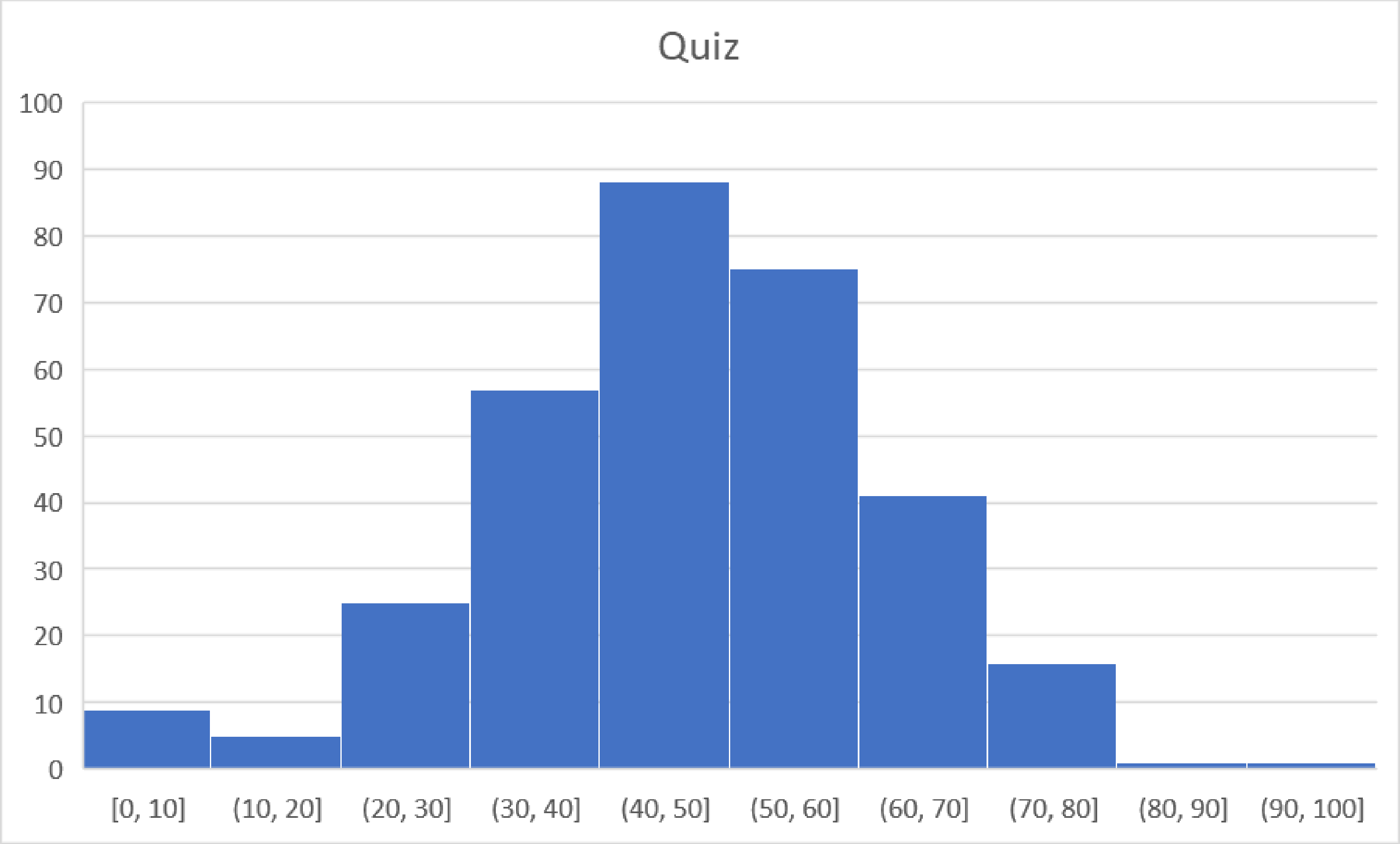
I would like to acknowledge the Traditional Owners of Australia and recognise their continuing connection to land, water and culture. I am currently on the land of the Darug people and pay my respects to their Elders, past, present and emerging.

I further acknowledge the Traditional Owners of the country on which you are on and pay respects to their Elders, past, present and future.

Copyright warning

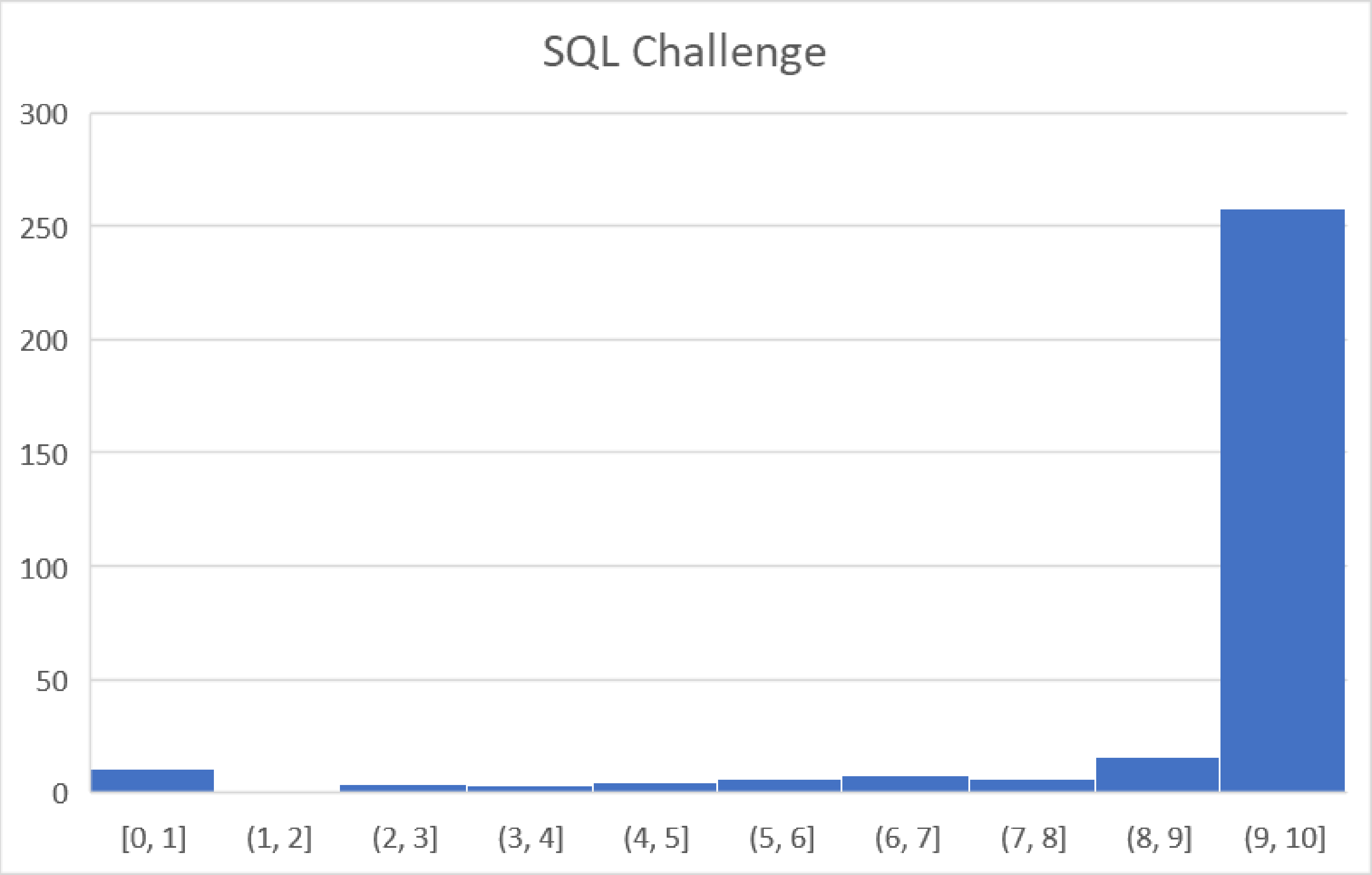
Quiz Summary

Average: 48%



SQL Challenge Summary

Average: 92%



Final Exam

- Date: Wednesday 16th November
- Time: 1:00 PM Sydney time
- Duration: 175 Minutes
  - Reading time: 10 Minutes
  - Writing time: 150 Minutes
  - Upload time: 15 minutes

- Everyone starts the exam at the same time
  - No late submission

- Exam adjustment is done by the exam office
  - Notification no later than 3 days before the exam

Keep track of your time during exam

THE UNIVERSITY OF SYDNEY

Final Exam

Question Type

- No MCQs
- Essay-type questions (10 Questions→50 marks)
  - ERD
  - Relational Model
  - Relational Algebra
  - SQL
  - Integrity Constraints
  - Transactions
  - Normalizations
  - Storage
  - Indexing
  - Query Processing

Practice exam is available in Canvas

› Exam covers 50% of the final mark

› You must obtain at least **40% in the exam**, as well as an **overall mark of at least 50%**, to pass the unit

THE UNIVERSITY OF SYDNEY

Course Outline

	Week	Topic
Foundations	Week 1	Introduction
	Week 2	Conceptual Database Design
	Week 3	Relational Data Model / Logical Database Design
	Week 4	Relational Algebra and SQL
	Week 5	Complex SQL
	Week 6	Database Integrity
Applications	~~Week 7~~	~~Database Application Development and Security~~
	Week 8	Transaction Management
	Week 9	Schema Refinement and Normalisation
Internals	Week 10	Storage and Indexing
	Week 11	Query Evaluation and Optimisation
	Week 12	Revision
	Week 13	Final exam Structure

DDL

```sql
CREATE TABLE Film(
        filmID  int PRIMARY KEY,
        title   varchar(30),
        releaseYear int);

CREATE TABLE Actor(
        actorID int PRIMARY KEY,
        actorName   varchar(30),
        nationality varchar(20));

CREATE TABLE FilmActor(
        filmID  int,
        actorID int DEFAULT 0,
        PRIMARY KEY(filmID, actorID),
        FOREIGN KEY (filmID) REFERENCES Film ON DELETE CASCADE,
        FOREIGN KEY (actorID) REFERENCES Actor ON UPDATE SET DEFAULT);
```

Film

filmID	title	releaseYear
101	No time to die	2021
102	Titanic	1998
205	Baby Day out	2000

Actor

actorID	actorName	nationality
0	No Actor	null
981	Daniel	UK
965	Kate	USA
901	Lily	Australia

FilmActor

filmID	actorID
101	981
102	965
205	901

What will be the records of Film and FilmActor table after executing the following query?

DELETE FROM Film **WHERE** filmID = 101;

Film

filmID	title	releaseYear
102	Titanic	1998
205	Baby Day out	2000

FilmActor

filmID	actorID
102	965
205	901

DDL

```sql
CREATE TABLE Film(
        filmID  int PRIMARY KEY,
        title   varchar(30),
        releaseYear int);

CREATE TABLE Actor(
        actorID int PRIMARY KEY,
        actorName   varchar(30),
        nationality varchar(20));

CREATE TABLE FilmActor(
        filmID  int,
        actorID int DEFAULT 0,
        PRIMARY KEY(filmID, actorID),
        FOREIGN KEY (filmID) REFERENCES Film ON DELETE CASCADE,
        FOREIGN KEY (actorID) REFERENCES Actor ON UPDATE SET DEFAULT);
```

Film

filmID	title	releaseYear
101	No time to die	2021
102	Titanic	1998
205	Baby Day out	2000

Actor

actorID	actorName	nationality
0	No Actor	null
981	Daniel	UK
965	Kate	USA
901	Lily	Australia

FilmActor

filmID	actorID
101	981
102	965
205	901

What will be the records of Actor and FilmActor table after executing the following query?

UPDATE Actor **SET** actorID = 999 **WHERE** actorID = 981;

Actor

actorID	actorName	nationality
0	No Actor	null
999	Daniel	UK
965	Kate	USA
901	Lily	Australia

FilmActor

filmID	actorID
101	0
102	965
205	901

DDL

```sql
CREATE TABLE Film(
        filmID  int PRIMARY KEY,
        title   varchar(30),
        releaseYear int);

CREATE TABLE Actor(
        actorID int PRIMARY KEY,
        actorName   varchar(30),
        nationality varchar(20));

CREATE TABLE FilmActor(
        filmID  int,
        actorID int DEFAULT 0,
        PRIMARY KEY(filmID, actorID),
        FOREIGN KEY (filmID) REFERENCES Film ON DELETE CASCADE,
        FOREIGN KEY (actorID) REFERENCES Actor ON UPDATE SET DEFAULT);
```

Film

filmID	title	releaseYear
101	No time to die	2021
102	Titanic	1998
205	Baby Day out	2000

Actor

actorID	actorName	nationality
0	No Actor	null
981	Daniel	UK
965	Kate	USA
901	Lily	Australia

FilmActor

filmID	actorID
101	981
102	965
205	901

- Create an assertion for the following:
  - An actor cannot take part in more than 3 movies in a year

```sql
CREATE ASSERTION ActingConstraint CHECK (
        NOT EXISTS (
                SELECT 1
                FROM Film NATURAL JOIN FilmActor
                GROUP BY actorID, releaseYear
                HAVING COUNT(filmID) > 3
        )
);
```

Review

› Consider the following relational schema:

Emp(*eid*, *ename*, *age*, *salary*)

Works(*eid*, *did*, *since*)

Dept(*did*, *budget*, *managerid*)

Define an assertion that will ensure that all managers have *age* > 30.

› **CREATE ASSERTION** managerAge

CHECK (**NOT EXISTS** (**SELECT** age

FROM Emp, Dept

WHERE eid = managerid **AND** age <= 30))

Review

Student(<u>*snum*</u>, *sname*, *major*, *level*, *age*)

Class(<u>*name*</u>, *meets_at*, *room*, *fid*)

Enrolled(<u>*snum*</u>, <u>*cname*</u>)

Faculty(<u>*fid*</u>, *fname*, *deptid*)

```
CREATE TABLE Enrolled (snum INTEGER, cname CHAR(20),
        PRIMARY KEY (snum, cname),
        FOREIGN KEY (snum) REFERENCES Student,
        FOREIGN KEY (cname) REFERENCES Class,
        CHECK (( SELECT COUNT (snum)
                FROM Enrolled
                GROUP BY cname) <= 30))
```

Express each of the following integrity constraints in SQL

- Every class has a maximum enrolment of 30 students.
- Every faculty member must teach at least two courses.
- No department can have more than 10 faculty members.

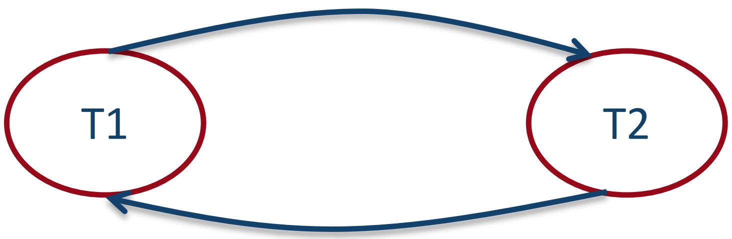
```
CREATE ASSERTION TeachConstraint
CHECK ( ( SELECT COUNT (*)
        FROM Faculty F NATURAL JOIN Class C
        GROUP BY C.fid
        HAVING COUNT (*) < 2) = 0)
```

```
CREATE TABLE Faculty (fid INTEGER, fname CHAR(20), deptid INTEGER,
        PRIMARY KEY (fnum),
        CHECK ( ( SELECT COUNT (*)
                FROM Faculty F
                GROUP BY F.deptid
                HAVING COUNT (*) > 10) = 0))
```

Review

Determine whether the following schedule is conflict serializable; justify your answer. If it is conflict serializable, please give a conflict equivalent serial schedule.

- R1(x),W2(y),R1(z),R3(z),W2(x),R1(y)



There are two conflicts in this case.

First conflict is between R1(x) and W2(x).

Because of this conflict we need to put T1 before T2 i.e T1 --->T2.

Second conflict is between W2(y) and R1(y).

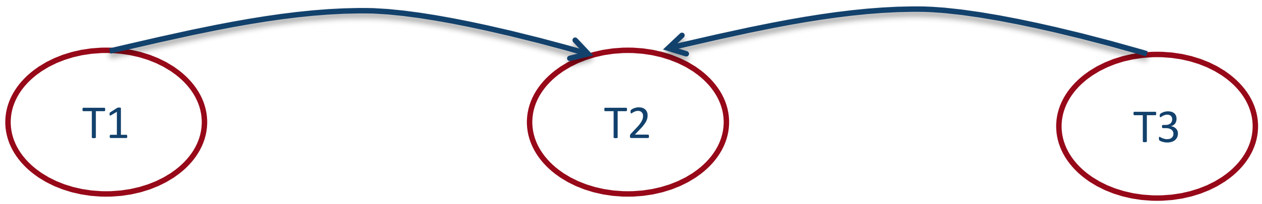
Because of this conflict we need to put T2 before T1 i.e T2 --->T1

Since we have a cycle between T1 and T2, that's why this schedule is NOT conflict serializable.

Review

Determine whether the following schedule is conflict serializable; justify your answer. If it is conflict serializable, please give a conflict equivalent serial schedule.

- R1(x),W2(y),R1(z),R3(x),W2(x),R2(y)



There are two conflicts in this case.

First conflict is between R1(x) and W2(x).

Because of this conflict we need to put T1 before T2 i.e T1 --->T2.

Second conflict is between R3(x) and W2(x).

Because of this conflict we need to put T3 before T2 i.e T3 --->T2

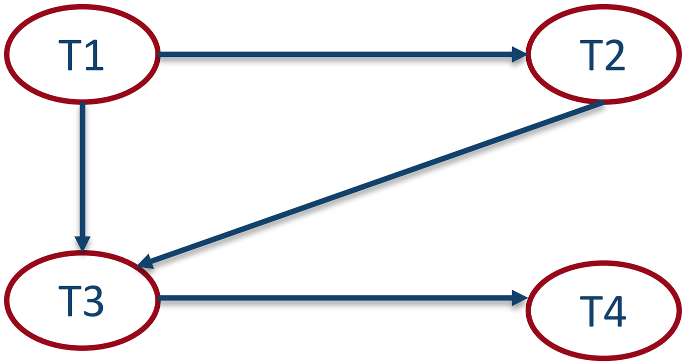
Since there is no cycle, so this schedule is conflict serializable.

According to the condition, T1 and T3 must be performed before T2. So, either of T1, T3, T2 OR T3, T1, T2 is a conflict equivalent serial schedule in this case.

Review

- Given the following schedule S:
  W1(x), R1(x), W3(z), R2(x), R2(y), R4(z), R3(x), W3(y)

Check whether this schedule is conflict serializable. If the schedule is conflict equivalent to a serial schedule, choose the equivalent serial order.



There are 4 conflicts in this case.

First conflict is between W1(x) and R2(x). Because of this conflict we need to put T1 before T2 i.e T1 --->T2.

Second conflict is between W1(x) and R3(x). Because of this conflict we need to put T1 before T3 i.e T1 --->T3

Third conflict is between W3(z) and R4(z). Because of this conflict we need to put T3 before T4 i.e T3 --->T4

Fourth conflict is between R2(y) and W3(y). Because of this conflict we need to put T2 before T3 i.e T2 --->T3

Since there is no cycle, so this schedule is conflict serializable. The equivalent serial order is T1, T2, T3, T4.

Review

Consider a database with a table Results(StudentId, UnitCode, Grade) that contains.

(Row)	StudentId	UnitCode	Grade
A	3245	INFO2120	57
B	3245	COMP2129	82
C	4290	INFO2120	68
D	4290	MATH2002	56

T1:	Begin Transaction
T1:	SELECT Grade FROM Results WHERE StudentId = '3245' AND UnitCode = 'COMP2129'
T2:	Begin Transaction
T2:	SELECT UnitCode, Grade FROM Results WHERE StudentId = '3245'
T2:	UPDATE Results SET Grade = Grade + 5 WHERE StudentId = '4290'
T1:	SELECT UnitCode, Grade FROM Results WHERE StudentId = '3245'
T1:	SELECT Sum(Grade) FROM Results WHERE UnitCode = 'INFO2120'
T1:	COMMIT
T2:	SELECT Sum(Grade) FROM Results WHERE UnitCode = 'INFO2120'
T2:	COMMIT

Figure 1

The steps in Figure 1 can be treated for the purposes of transaction management as an execution schedule consisting of read (r) and write (w) operations by either transaction 1 or 2 upon different objects - in this case we shall assume that the granularity of these objects is row-level, with each row denoted by the letter given in the table above.

Which of the following execution schedule matches the steps in Figure 1?

1. r1(B),r2(A),r2(B),r2(C),r2(D),r1(A),r1(B),r1(A),r1(C),r2(A),r2(C)
2. r1(B),r2(A),r2(B),w2(C),w2(D),r1(A),r1(B),r1(A),r1(C),r2(A),r2(C)
3. r1(B),r2(A),r2(B),r2(C),w2(C),r2(D),w2(D),r1(A),r1(B),r1(A),r1(C),r2(A),r2(C)

Which FD might be valid?

A	B	C	D
1	1	2	a
1	2	1	b
2	2	3	c
1	2	3	d
3	2	1	e
1	2	1	f

1. ~~AB → C~~
2. AC → B
3. ~~BC → A~~
4. ~~A → BC~~
5. ~~B → AC~~

Review

- Consider the following instance of a relation.

A	B	C	D
1	x	a	10
2	y	b	20
3	z	b	20
1	z	c	30
2	y	b	20

- Which of the following functional dependencies might be valid for this relation?

1. AB -> C
2. C -> D
3. A -> B
4. ABC -> D
5. B -> C
6. D -> C

Review

Consider attributes of the relation, describing the cost of damage to people who have been involved in car accidents:

Damage(regno, license, address, accidentID, amount)

Match each of the business rules below with the corresponding functional dependency.

A. A car (regno) can have at most one driver (license) **4**

B. Drivers must be registered at a single address **5**

C. A driver can be in at most one car in any given accident **6**

Functional Dependencies

1. regno, accidentID -> license
2. license -> regno
3. address -> license
4. regno -> license
5. license -> address
6. license, accidentID -> regno

Review

$R(A, B, C, G, H, I)$

$F = \{$

$A \rightarrow B$

$A \rightarrow C$

$CG \rightarrow H$

$CG \rightarrow I$

$B \rightarrow H$

$\}$

Is *(AG) a candidate key?*

› Compute the attribute closure $(AG)^+$

1. *result = AG*
2. *result = ABG* $(A \rightarrow B)$
3. *result = ABCG* $(A \rightarrow C)$
4. *result = ABCGH* $(CG \rightarrow H)$
5. *result = ABCGHI* $(CG \rightarrow I)$

› Is *(AG) a candidate key?*

1. Is (AG) a super key? YES! The closure $(AG)^+ = R$ *(i.e., **all** attributes in R)*
2. Is (AG) minimal? (Is any subset of (AG) a superkey?)
   - $(A)^+ = ABCH \neq R$
   - $(G)^+ = G \neq R$

Therefore AG is a candidate key

THE UNIVERSITY OF SYDNEY

Review

- Consider a relation R(A, B, C, D, E, F) with functional dependencies:

  A -> BDF

  BC -> DEF

  D -> CE.

- Is AB a candidate key ?
- Compute the closure *(AB)*$^+$
  1. *result = AB*
  2. *result = ABDF (A → BDF)*

     = ABCDEF (D → CE)
- *AB is a superkey because the closure is R*
- Compute the closure *A*$^+$
  1. *result = A*
  2. *result = ABDF (A → BDF)*

     = ABCDEF (D → CE)
- *AB is not a candidate key because a subset of (AB), i.e. A, is a key.*

Review

Work on a project is tracked with the following relation:

WorksOn (EmpID, ProjID, start, finish, manager)

The following FDs hold over this relation:

EmpID -> manager	(Each employee can only have one manager)
EmpID, ProjID -> start, finish	(Each employee can only work on a project once)
ProjID, start -> EmpID	(Only one employee can start work on the project at a time)

Which of the following are candidate keys for the relation?

1. **ProjID, start**
2. **EmpID, ProjID**
3. **start, finish, manager**
4. **EmpID, ProjID, start**

Review

Based on the data in the relation, is this a valid MVD?

$UoS \twoheadrightarrow Lecturer$

Note that according to the values in the relation, the relationship between the UoS and Lecturer is ***independent*** from the relationship between UoS and Textbook. *This means that the above MVD is valid*. This also implies that the Lecturer of a UoS is selected independently by the school.

Assume a new lecturer, Lijun C is added for the UoS COMP9120. What must happen to maintain this independence (i.e., MVD)?

- Add one row for each different textbook with Lecturer *Lijun C* of that UoS.

UoS	Textbook	Lecturer
COMP9120	Silberschatz	Ying Z
COMP9120	Widom	Ying Z
COMP9120	Silberschatz	Mohammad P
COMP9120	Widom	Mohammad P
COMP9120	Silberschatz	Alan F
COMP9120	Widom	Alan F
COMP5110	Silberschatz	Ying Z
COMP5110	Silberschatz	Mohammad P
COMP9120	**Widom**	**Lijun C**
COMP9120	**Silberschatz**	**Lijun C**

Review

Assume the only key to the following relation is the set (UoS, Textbook, Lecturer).

Is this relation in 4NF?

UoS	Textbook	Lecturer
COMP9120	Silberschatz	Ying Z
COMP9120	Widom	Ying Z
COMP9120	Silberschatz	Mohammad P
COMP9120	Widom	Mohammad P
COMP9120	Silberschatz	Alan F
COMP9120	Widom	Alan F
COMP5110	Silberschatz	Ying Z
COMP5110	Silberschatz	Mohammad P

No: There are at least *two non-trivial multivalued dependencies*

UoS ↠ Textbook *and* UoS ↠ Lecturer

and UoS is *not* a superkey.

Solution: Split the above relation into two relations:

UoS	Lecturer

UoS	Textbook

Now both relations are

In 4NF! Why?

MVDs are trivial now!

Review

uosCode	lecturerId
COMP5138	3456
COMP5338	4567

```
BEGIN;
UPDATE Course SET lecturerId=4567 WHERE uosCode='COMP5138';
COMMIT;
SELECT lecturerId FROM Course WHERE uosCode='COMP5138';
```

1. 1234
2. 3456
3. 4567

Review

- Give a set of FDs for the relation schema *R(A,B,C,D)* with candidate key *AB* so that *R* is not in 2NF.

2NF: *There cannot be a functional dependency between a subset of a key to non-key attributes*. This is applicable only when the key consists of more than one attribute.

An example of such FDs is:

$AB \rightarrow CD$

$B \rightarrow C$

AB is obviously a key for this relation since $AB \rightarrow CD$ implies $AB \rightarrow ABCD$. However, the FD $B \rightarrow C$ violates 2NF since:

B is a proper subset of the key *AB*

C is not part of a key

- Give a set of FDs for the relation schema *R(A,B,C,D)* with candidate key *AB* under which *R* is in 2NF but not in 3NF.

Third Normal Form (3NF)

Formal Definition: a relation R is in 3NF if for each dependency X→ Y in F^+, *at least one of the following holds*:

X → Y is a trivial FD (Y ⊆ X)

X is a superkey for R

Y ⊂ (is a proper subset of) a candidate key for R

An example of such FDs is:

AB → CD

C → D

AB is obviously a key for this relation since *AB → CD* implies *AB → ABCD*. The FD: *C → D* violates 3NF but complies with 2NF since:

C → D is not a trivial FD

C is not a superkey

D is not part of some key for *R*

- Consider a relation *R* with attributes *ABCDE* and the following FDs:

  $A \rightarrow BC$, $BC \rightarrow E$, and $E \rightarrow DA$.

Is *R* in BCNF? If not, give a lossless join decomposition of R.

The schema R has keys *A*, *E* and *BC*. Why? Use attribute closure to find keys.

It follows that R is in BCNF.

- Let *S* be a relation with attributes *ABCDE* and the following FDs are given:

  $A \rightarrow CE$, $D \rightarrow B$, and $E \rightarrow DA$.

Is S in BCNF? If not, give a lossless join decomposition of R.

The schema S has keys *A* and *E*. Use attribute closure to find keys.

It follows that S is *not* in BCNF.

Therefore, decompose S into S1 = (D,B) and S2 = (A, C, D, E). It is lossless join decomposition because the intersection of S1 and S2 is a key to S1.

Review

Is the following relation in BCNF? If not, give a lossless-join decomposition.

***contracts* (*contractID, supplierID, projectID, deptID, itemID, quantity, value*)**

Functional dependencies:

contractID → supplierID, projectID, deptID, itemID, quantity, value
supplierID, deptID → itemID
projectID → supplierID

Solution:
It is not in BCNF. Because in the given 2^{nd} and 3^{rd} FDs, the LHS is not a key. How do we know?
Use attribute closure.

Therefore, looking at **supplierID, deptID → itemID**, we can divide *contracts* relation into

R1 = (supplierID, deptID, itemID) and
R2 = (supplierID, deptID, contractID, projectID, quantity, value)

Then looking at **projectID → supplierID**, we can divide R2 into

R3 = (projectID , supplierID) and
R4 = (projectID, deptID, contractID, quantity, value)

Therefore, R1, R3 and R4 is the lossless-join decomposition of *contracts* relation. Note the intersection of the above relations is a key to one of the relations.

Review

Assume that there are 1,000,000 records in a relation R and each record is 200 bytes long. Each page is 4K bytes, of which 250 bytes are reserved for header and array of record pointers. Assume pages are 90% full on average.

- How many records can we store in each page?
- How many pages are required to store R?

Solution:

Empty space in each page is (4*1024 – 250) = 3846 bytes

Number of records per page = floor(3846 / 200) = 19 records

On average, each page contains floor(19 * 90%) = 17 records

Number of pages required to store the table = ceil(1,000,000/17) = 58,824 pages

Query Processing

- True/False?
  - Parser takes a query-evaluation plan, executes that plan, and returns the answers. False
  - SQL systems remove duplicates even if the keyword **DISTINCT** is not specified in a query. False
  - Pipelined evaluation is cheaper than materialization True
  - External merge-sort works even if the entire table does not fit in the main memory True
  - Natural join is a special case of Equi-join True

Join

Consider two tables employee(eid, ename, did) and department(did, dname). There are 10000 tuples in the employee table and 1000 tuples in the department table. It requires 500 pages and 100 pages to store the records of employee and department table respectively.

› Calculate the cost of nested loop join and block-nested loop join of these two tables.

The estimated cost of nested loops join

employee as outer table: 500+ 10000 * 100 = **1,000,500 disk I/Os**

department as outer table: 100 + 1000 ∗ 500 = **500,100 disk I/Os**

The estimated cost of block nested loops join

employee as outer table: 500 + 500 * 100 = **50,500 disk I/Os**

department as outer table: 100 + 100 ∗ 500 = **50,100 disk I/Os**

That’s it for today..

See you next week!