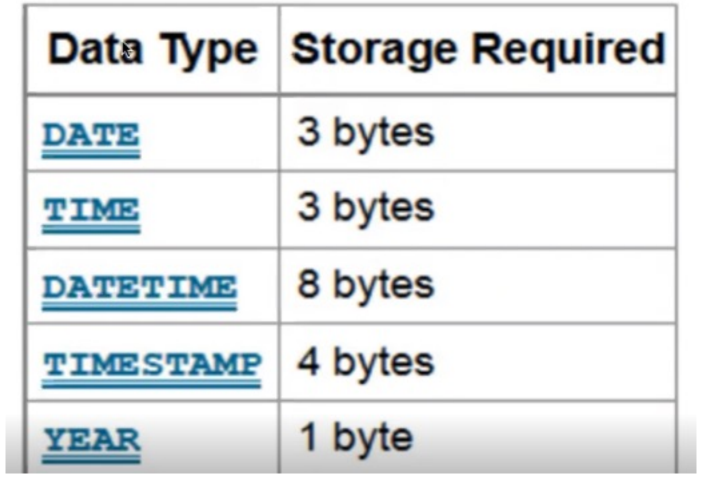
INFO20003 Database Systems

Week 10

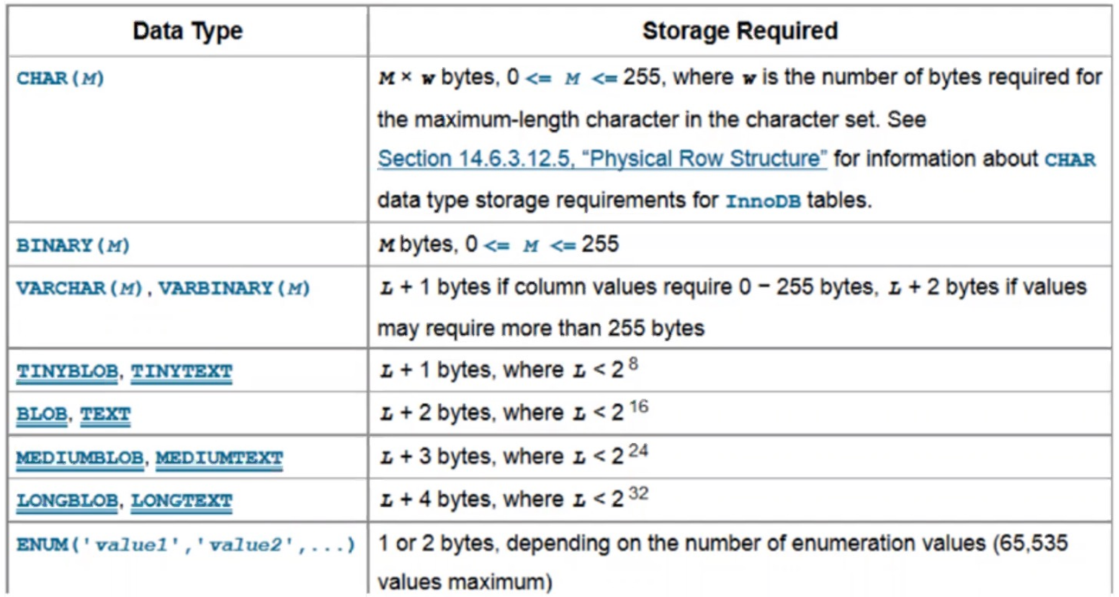
Capacity Planning

Capacity Planning

- Estimate the size of database by adding up table sizes
- Table size = nRows x average row length
- Average row length = sum of attribute type sizes
- Example type sizes:



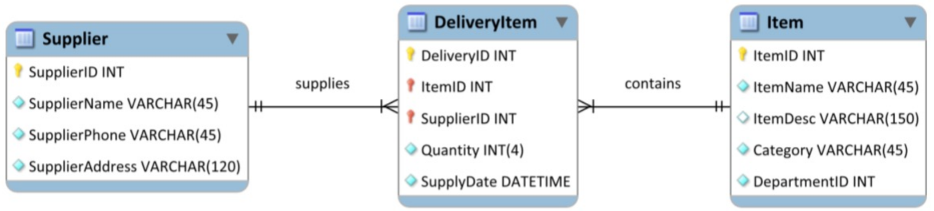
Data Type	Storage Required
DATE	3 bytes
TIME	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
YEAR	1 byte



Data Type	Storage Required
CHAR(*M*)	*M* × *w* bytes, 0 <= *M* <= 255, where *w* is the number of bytes required for the maximum-length character in the character set. See Section 14.6.3.12.5, "Physical Row Structure" for information about CHAR data type storage requirements for InnoDB tables.
BINARY(*M*)	*M* bytes, 0 <= *M* <= 255
VARCHAR(*M*), VARBINARY(*M*)	*L* + 1 bytes if column values require 0 − 255 bytes, *L* + 2 bytes if values may require more than 255 bytes
TINYBLOB, TINYTEXT	*L* + 1 bytes, where $L < 2^8$
BLOB, TEXT	*L* + 2 bytes, where $L < 2^{16}$
MEDIUMBLOB, MEDIUMTEXT	*L* + 3 bytes, where $L < 2^{24}$
LONGBLOB, LONGTEXT	*L* + 4 bytes, where $L < 2^{32}$
ENUM('*value1*','*value2*',...)	1 or 2 bytes, depending on the number of enumeration values (65,535 values maximum)

Q1) Capacity Planning

1. **Capacity planning**



Consider the case of a department store. An analyst has determined that there are 50 distinct suppliers that provide 2000 distinct items to the store. They have determined that the average delivery is of 40 distinct items and that each supplier delivers approximately once a week (the analyst has estimated this to be 50 deliveries a year). For each delivery by a supplier, there are on average 40 rows added to the DeliveryItem table. While the Item and Supplier tables stay constant in size, the DeliveryItem table grows by 100,000 rows every year.

This assumes that suppliers and items stay constant; however, if the business is successful, the suppliers and frequency of deliveries and number of distinct items delivered can be expected to grow. If we know the length of each row, we can estimate how big each table will be year by year.

Using information about data type storage from the MySQL documentation and information from the data dictionary, the analyst has determined the following average row lengths of each table:

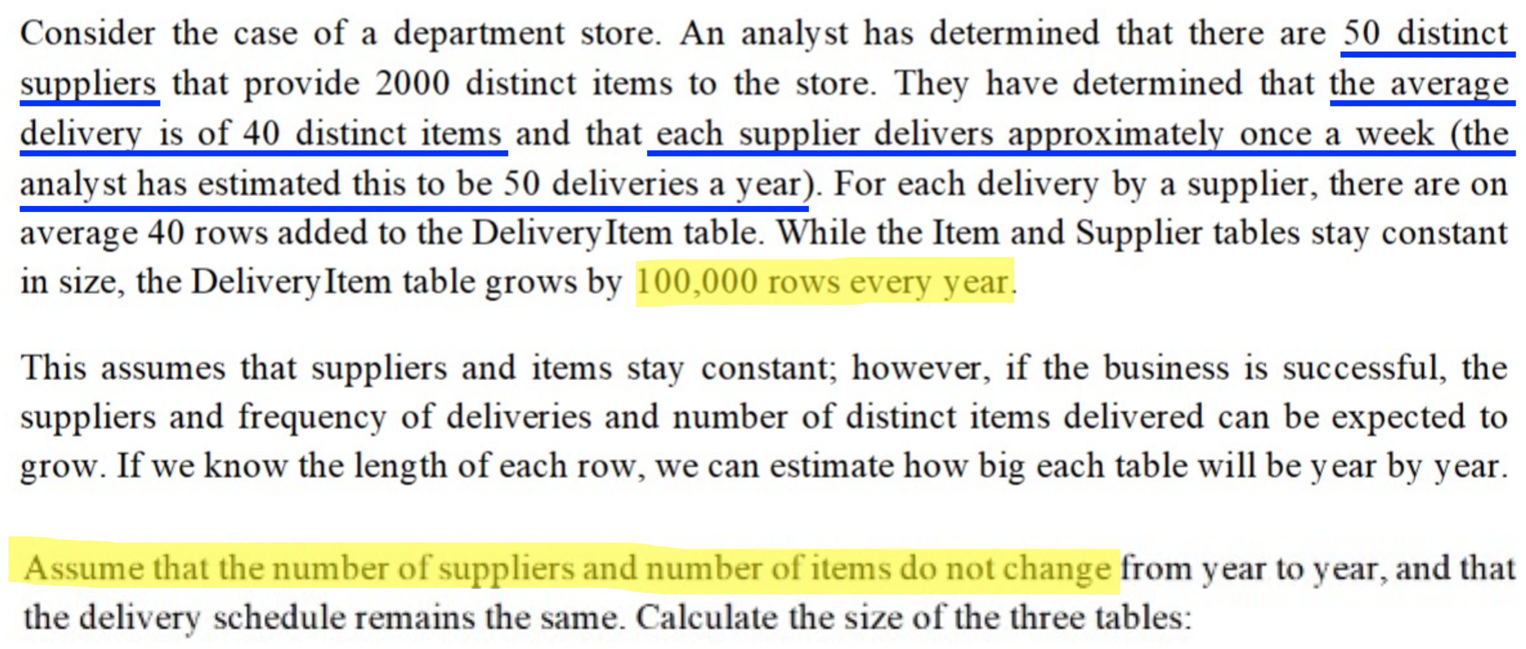
Table	**Number of rows**	**Average row length**
Supplier	50 rows	144 bytes
Item	2000 rows	170 bytes
DeliveryItem	0 rows	19 bytes

Table 1: Row volume and row length for the Supplier delivers Item entities.

Assume that the number of suppliers and number of items do not change from year to year, and that the delivery schedule remains the same. Calculate the size of the three tables:

a. When database use begins (year 0)
b. After one year of database use
c. After five years of database use

Q1) Capacity Planning



Consider the case of a department store. An analyst has determined that there are 50 distinct suppliers that provide 2000 distinct items to the store. They have determined that the average delivery is of 40 distinct items and that each supplier delivers approximately once a week (the analyst has estimated this to be 50 deliveries a year). For each delivery by a supplier, there are on average 40 rows added to the DeliveryItem table. While the Item and Supplier tables stay constant in size, the DeliveryItem table grows by 100,000 rows every year.

This assumes that suppliers and items stay constant; however, if the business is successful, the suppliers and frequency of deliveries and number of distinct items delivered can be expected to grow. If we know the length of each row, we can estimate how big each table will be year by year.

Assume that the number of suppliers and number of items do not change from year to year, and that the delivery schedule remains the same. Calculate the size of the three tables:

50 suppliers	x	50 deliveries/supplier	=	2500 deliveries/yr
2500 deliveries/yr	x	40 items/delivery	=	100,000 deliveryItems/yr

Q1)

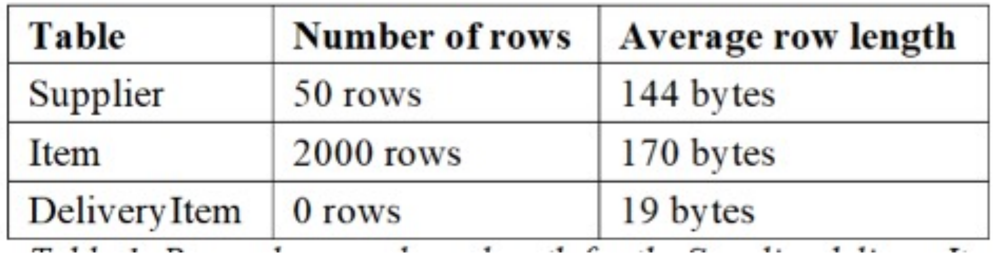


Table	Number of rows	Average row length
Supplier	50 rows	144 bytes
Item	2000 rows	170 bytes
DeliveryItem	0 rows	19 bytes

Constant
Constant
100,000 rows per yr

a. When database use begins (year 0)

Supplier = 50 × 144 bytes = 7200 bytes (approx. 7 KB – *Note: 1 KB = 1024 bytes*)
Item = 2000 × 170 bytes = 340,000 bytes (approx. 332 KB)
DeliveryItem = 0 × 19 bytes = 0 bytes

b. After one year of database use

Supplier and Item remain unchanged.
DeliveryItem = 100,000 × 19 bytes = 1,900,000 bytes (approx. 1.8 MB – *Note: 1 MB = 1024 KB = 1,048,576 bytes*)

c. After five years of database use

Supplier and Item remain unchanged.
DeliveryItem = 500,000 × 19 bytes = 9,500,000 bytes (approx. 9.1 MB)

Backup

Backup - Motivation

- To recover in the event of failure of the database system
- Avoid:
  - loss of data
  - data integrity errors
  - data mismatch

Backup Type

- Logical
- Physical

Logical Backup

- Backup completed through **SQL queries**
- keep a record of the data, and all the metadata
- take more information than physical backups
- Larger and slower than physical backup

Advantage:

- to move data from one operating system to another
- migrating data from one database to a completely different database and environment

CANNOT BE OFFLINE BACKUP

Physical Backup

- a direct image copy of the physical database files on the disk
- fastest way to make a copy of the database

- In the event of a database failure:
  - the **physical copies** can be restored to their original location
  - DBA replay all the transactions using the **Crash Recovery log**

- Either **online or offline**

Backup mode

Online

- the users are still connected and are unaffected by the backup operations

Offline

- the database server process is shut down while the physical copy of the file is made

Backup Location

Onsite

- stored on the same premises – but not the same machine as the database

Offsite

- stored in a remote location (>160 km away)

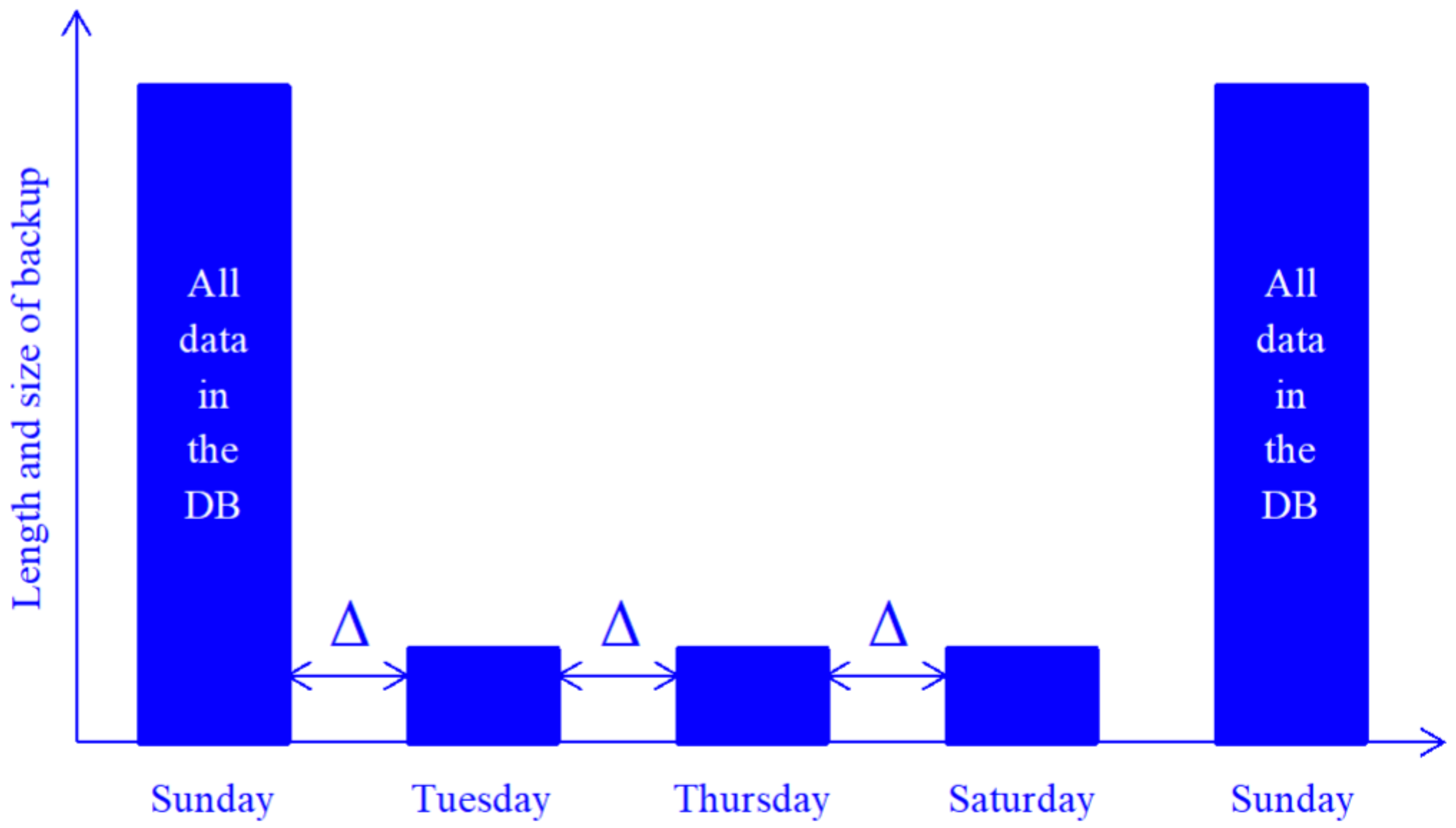
Full backups and incremental backups

Full

- back up all data in our database

Incremental

- only backs up the changes since the last backup
- smaller in size and shorter in duration


Length and size of backup
All data in the DB
Δ
Δ
Δ
All data in the DB
Sunday
Tuesday
Thursday
Saturday
Sunday

Data recovery

- restore the backup
- recover to the point of failure using the recovery log

Q2)

2. Backup and recovery case study

ACME Manufacturing makes widgets in its factory. The factory runs 24 hours a day, 7 days a week in three shifts. The quietest shift is the Sunday night shift, which runs from midnight Sunday to 8am Monday. While ACME manufactures widgets, the database must run. This is ACME's only widget factory.

The database administrator has implemented a backup policy that takes a full backup every Sunday at 3am during the night shift, and then an incremental backup on Tuesday, Thursday and Saturday mornings at 3am.

The backup strategy has determined that if there is a database failure, restoration of the database is time-critical. ACME must have the shortest outage time to restore and recover the database. This means the database must be restored quickly so that the manufacturing can continue. ACME must have the smallest elapsed time from the point of failure to the database being fully operational and useable.

a. Given the business requirements and the database administrator's backup policy, what database backup type, mode and site would you recommend?

b. Consider the Full and Incremental backup timeline in Figure 1. If the database suffered a media failure on Friday at 9:23am, how many backups would need to be restored?

c. Given the same failure, what would be the benefits and costs of changing the backup strategy to do full backups on Sunday, Tuesday, Thursday and Saturday mornings at 3am?

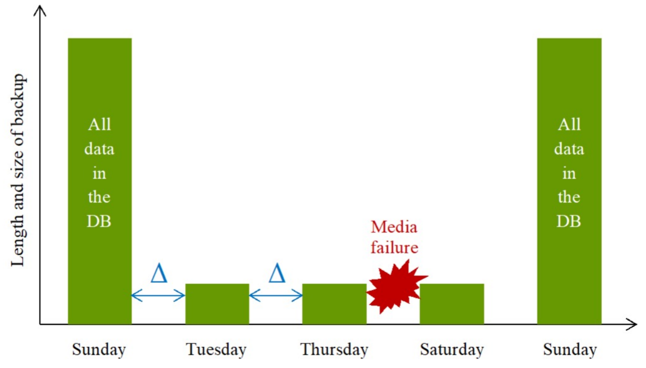
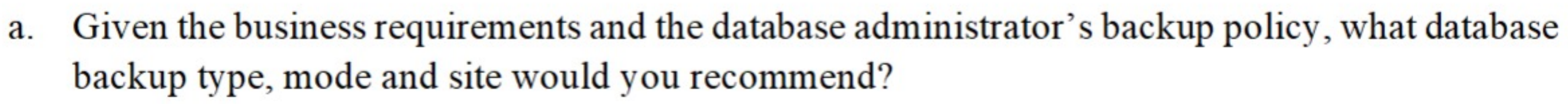
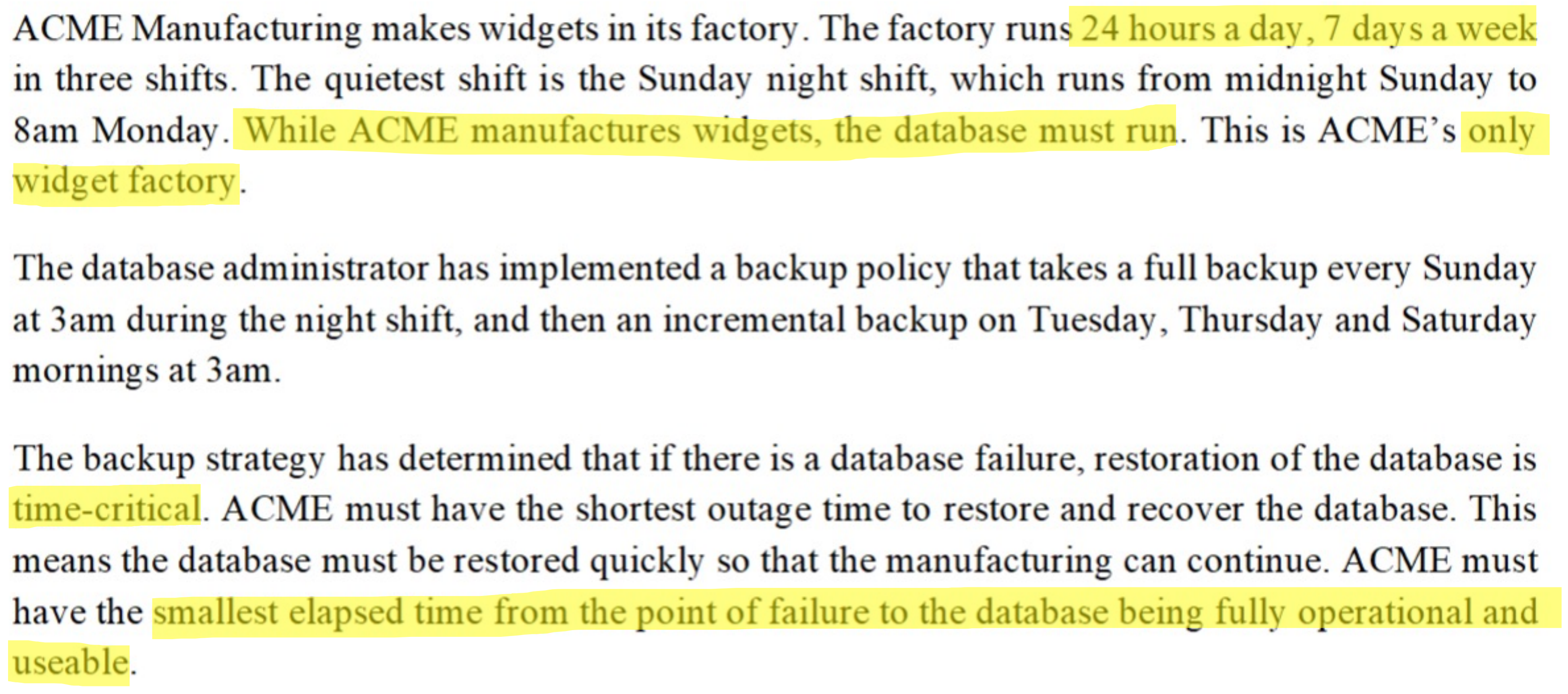


Figure 1. A timeline of full and incremental backups showing the media failure on Friday morning.

Q2a)


a. Given the business requirements and the database administrator's backup policy, what database backup type, mode and site would you recommend?


ACME Manufacturing makes widgets in its factory. The factory runs 24 hours a day, 7 days a week in three shifts. The quietest shift is the Sunday night shift, which runs from midnight Sunday to 8am Monday. While ACME manufactures widgets, the database must run. This is ACME's only widget factory.

online

onsite

The database administrator has implemented a backup policy that takes a full backup every Sunday at 3am during the night shift, and then an incremental backup on Tuesday, Thursday and Saturday mornings at 3am.

The backup strategy has determined that if there is a database failure, restoration of the database is time-critical. ACME must have the shortest outage time to restore and recover the database. This means the database must be restored quickly so that the manufacturing can continue. ACME must have the smallest elapsed time from the point of failure to the database being fully operational and useable.

physical

Q2b)

b. Consider the Full and Incremental backup timeline in Figure 1. If the database suffered a media failure on Friday at 9:23am, how many backups would need to be restored?

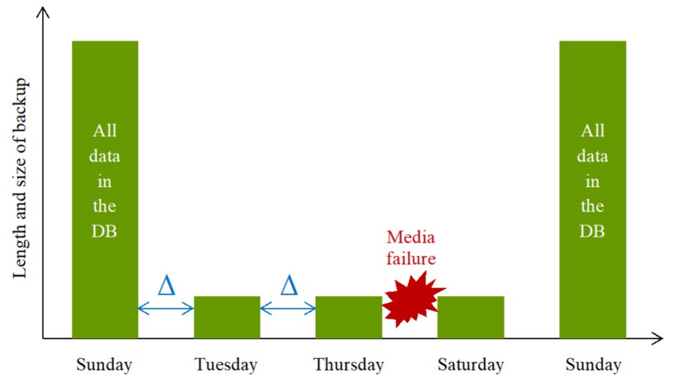


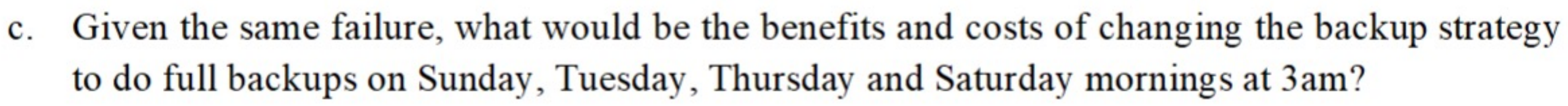
Figure 1. A timeline of full and incremental backups showing the media failure on Friday morning.

Three backups would need to be restored:

- The Sunday full backup
- The Tuesday incremental backup
- The Thursday incremental backup

The DBA would also replay the crash recovery logs from Thursday morning until Friday 9:23am.

Q2c)



c. Given the same failure, what would be the benefits and costs of changing the backup strategy to do full backups on Sunday, Tuesday, Thursday and Saturday mornings at 3am?

Benefits

- reduce the time to fully restore the database

Costs

- More space would be required
- Backup will take longer to complete
- There is a risk that if we don’t remove the backups from the database server, we could fill up the file system

Transaction

Transaction

- a logical unit of work that must either be **entirely completed** or **aborted**

- usually corresponds to a single “action” that involves several changes to the database
  - E.g. removing an employee and all their related data from the database
  - E.g. an actual financial transaction

- adhere to the ACID principles

ACID

Atomic – Each transaction either entirely succeeds or entirely fails. If a failure occurs mid-transaction, the DBMS must discard (roll back) all of the transaction's changes up to that point.

Consistent – Upon completion, a transaction must respect data integrity rules (constraints) and leave the database in a consistent state.

Isolated – Changes made in one transaction cannot be seen from within other transactions.

Durable – Once a transaction is committed, the inserts, updates and deletes carried out in that transaction persist permanently in the database.

Concurrency

- the ability to allow many users to connect to and work with the database simultaneously

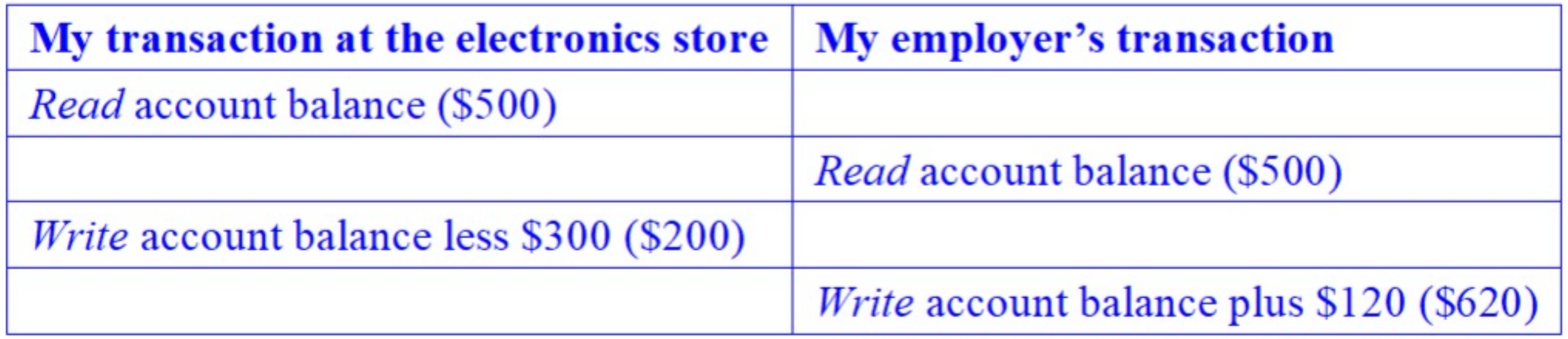
Concurrency Problems

- Lost Update Problem
- Uncommitted Data Problem
- Inconsistent Retrieval Problem

Lost Update Problem

- When two users attempt to update the same piece of data at the same time
- The second user ignores the update made by the first user

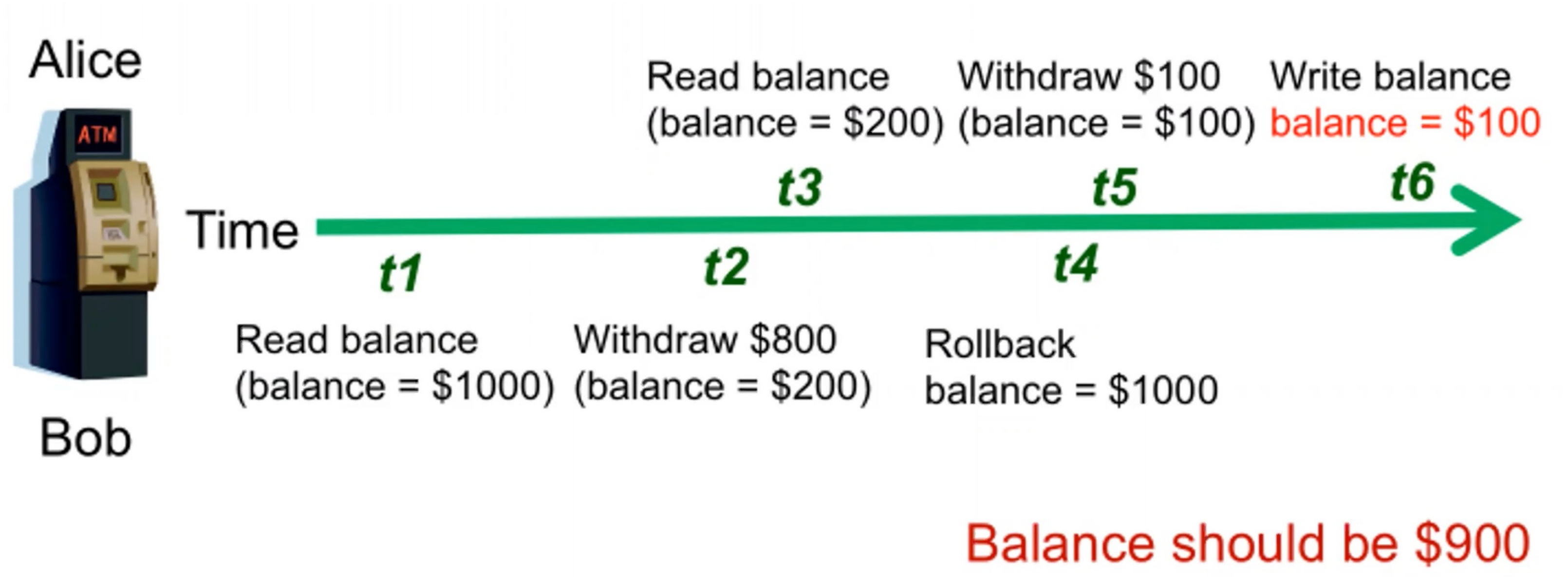
- E.g. my bank account contains $500. I am buying a $300 TV, while at exactly the same time, my employer is paying me $120. Instead of being $180 poorer, I am now $120 richer!



My transaction at the electronics store	**My employer's transaction**
Read account balance ($500)	
	Read account balance ($500)
Write account balance less $300 ($200)	
	Write account balance plus $120 ($620)

Uncommitted Data Problem

- User1 updates a value, which is used by user2, but then user1 rolls back
- Violates Isolated property of transaction


Alice
Read balance (balance = $200)
Withdraw $100 (balance = $100)
Write balance balance = $100
Time
t1
t2
t3
t4
t5
t6
Read balance (balance = $1000)
Withdraw $800 (balance = $200)
Rollback balance = $1000
Bob
Balance should be $900

Inconsistent Retrieval Problem

- User1 runs a query over a large amount of data, while user2 is making changes to some data.
- User1 gets some modified data, some un-modified data

Alice	Bob

SELECT SUM(Salary)
FROM Employee;
	UPDATE Employee
	SET Salary = Salary * 1.01
	WHERE EmpID = 33;

	UPDATE Employee
	SET Salary = Salary * 1.01
	WHERE EmpID = 44;

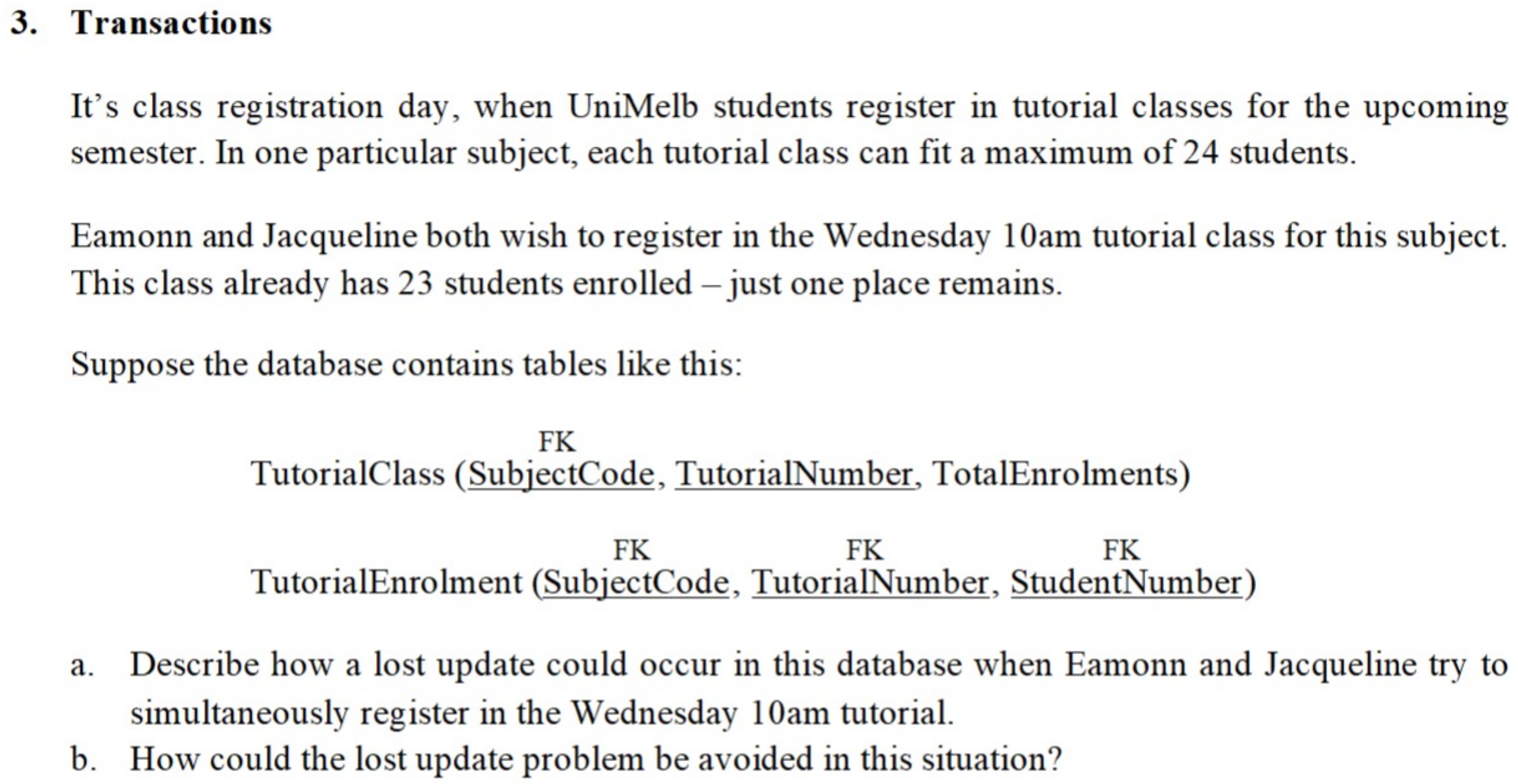
(finishes calculating sum)	COMMIT;

Red = Bob's action (updating salaries)
Black = Alice's action (calculating sum of salary)

Time	Trans-action	Action	Value	T1 SUM	Comment
1	T1	Read Salary for EmpID 11	10,000	10,000	
2	T1	Read Salary for EmpID 22	20,000	30,000	
3	T2	Read Salary for EmpID 33	30,000		
4	T2	Salary = Salary * 1.01			
5	T2	Write Salary for EmpID 33	30,300		
6	T1	Read Salary for EmpID 33	30,300	60,300	*after* update
7	T1	Read Salary for EmpID 44	40,000	100,300	*before* update
8	T2	Read Salary for EmpID 44	40,000		
9	T2	Salary = Salary * 1.01			
10	T2	Write Salary for EmpID 44	40,400		
11	T2	COMMIT			
12	T1	Read Salary for EmpID 55	50,000	150,300	
13	T1	Read Salary for EmpID 66	60,000	210,300	

we want either
before $210,000 or
after $210,700

Q3) Transaction



3. Transactions

It's class registration day, when UniMelb students register in tutorial classes for the upcoming semester. In one particular subject, each tutorial class can fit a maximum of 24 students.

Eamonn and Jacqueline both wish to register in the Wednesday 10am tutorial class for this subject. This class already has 23 students enrolled – just one place remains.

Suppose the database contains tables like this:

$$\text{TutorialClass}\ (\overset{\text{FK}}{\underline{\text{SubjectCode}}},\ \underline{\text{TutorialNumber}},\ \text{TotalEnrolments})$$

$$\text{TutorialEnrolment}\ (\overset{\text{FK}}{\underline{\text{SubjectCode}}},\ \overset{\text{FK}}{\underline{\text{TutorialNumber}}},\ \overset{\text{FK}}{\underline{\text{StudentNumber}}})$$

a. Describe how a lost update could occur in this database when Eamonn and Jacqueline try to simultaneously register in the Wednesday 10am tutorial.

b. How could the lost update problem be avoided in this situation?

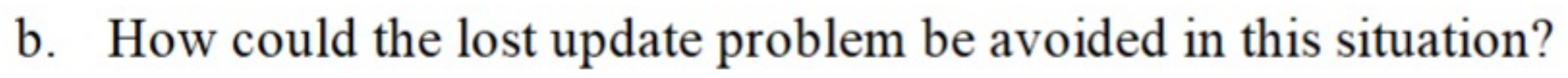
Q3a)

a. Describe how a lost update could occur in this database when Eamonn and Jacqueline try to simultaneously register in the Wednesday 10am tutorial.

Eamonn	**Jacqueline**
Read TutorialClass.TotalEnrolments (23)	
	Read TutorialClass.TotalEnrolments (23)
Insert row into TutorialEnrolment	
	Insert row into TutorialEnrolment
	Write TutorialClass.TotalEnrolments (24)
Write TutorialClass.TotalEnrolments (24)	

Even though there are now 25 students enrolled in the class, the value of TotalEnrolments for this class is equal to 24. A lost update has occurred.

Q3b)



b. How could the lost update problem be avoided in this situation?

- Serial execution
  - Inefficient
  - Guarantee that isolation is satisfied
- Locking
  - More efficient
- Timestamp
- Optimistic concurrency control