(أهلا وسهلا) ←

Contoh **tajuk dialog,**

مَادَّة : اللغة العربية TAC401

(subjek)

كُلِّيَّة : العلوم التطبيقية

(fakulti)

مَجْمُوعَة : AST1AA

(kump)

مُحَاضِرَة : أستاذة أسماء بنت عمَّار

(pensyarah)

فَصْل دِرَاسِيّ 2020

(sem/penggal)

MUKA HADAPAN SKRIP		
	TAC401	HIJAU
	TAC451	UNGGU
	TAC501	BIRU

أَعْضَاء المَجْمُوعَة

(anggota kumpulan)

فاطمة بنت إسماعيل 3390202-013

شفيق بن عبد الله 8768680-014

حسن بن عثمان 2292292-012

ISTILAH YANG BOLEH DIGUNAKAN

Para Pelakon	مُمَثِّلُون
:Penghargaan khas	إِهْدَاء خَاص :
….Kami Mempersembahkan	نُقَدِّم ...
Babak/scene 1	مَشْهَد 1
Script	نَصّ
Penerbit	مُخْرِج
Bunyi	صَوْت
scriptwriter	كِاتِب السِّينَارِيُو
flashback	الارْتِجَاع الفَنِّي
subtitle	الحَاشِيَة السِّينِيمَائِيَّة
Montage	مُونْتَاج
Pelakon tambahan	مُمَثِّل إضَافِيّ
Tamat/the end	نِهَايَة

GARIS PANDUAN PENYEDIAAN SKRIP BAHASA ARAB

- Penggunaan *google translation* dan lain-lain perlu merujuk kepada pensyarah.
- Letakkan gambar ahli dan nama penuh pada muka surat pertama selepas cover utk membantu pensyarah mengenal pasti watak ketika sesi lakonan berlangsung.
- Guna perkataan/istilah/panduan yang disediakan (nama kumpulan/kumpulan dll) untuk muka depan skrip.
- Mempunyai pendahuluan dan penutup.
- Menggunakan karakter arab dalam penulisan skrip

- Menggunakan *font* Traditional Arabic bersaiz 18 untuk *title* dan saiz 16 untuk *content*
- Menggunakan *spacing* 1.5 bagi setiap ayat

RUJUKAN - WEBSITE ARABIC SOFTWARE:

http://mprofaca.cro.net/arabic_keyboard.html

http://www.arabic-keyboard.org/

http://www.indiana.edu/~arabic/typing_arabic.htm

http://www.freedownloadscenter.com/Games/Educational_Games/Arabic_Keyboard_Typing_Tutor.html

http://www.download3000.com/download_12554.html

http://3d2f.com/programs/119-751-arabic-keyboard-typing-tutor-download.shtml

http://store.aramedia.com/product.php?xProd=6226

PANDUAN PENYEDIAAN LAPORAN (UNTUK SEMAKAN)

BIL	PERLU DISEDIAKAN	TANDA (√) SETELAH TINDAKAN DILAKUKAN
1	Cover /logo uitm/nama pensyarah/group arab/nama setiap pelajar	
2	Pembukaan Cth: babak pertama : nyatakan tajuk/lokasi berlakon/siapa pelakon-pelakon	
3	Gambar pelajar/ nombor matrik pelajar/nombor telefon.	
4	Skrip dialog/bertaip arab/terjemahan B.M atau B.I. Nama pelakon mesti nama asal pelajar yang bertaip penuh dan bukannya nama panggilan atau samaran.	
5	Penutup dan penghargaan	
6	Cover mengikut warna yang di tetapkan dan di*binding.*	
7	Dihantar bersama ketika sesi lakonan	
8	Penghantaran lewat akan ditolak markah	

BORANG AKUAN PEYERAHAN SKRIP

NAMA KUMPULAN… TARIKH: ……

Saya nama ……, No pelajar: ……..…… merupakan ketua kumpulan bagi semua ahli berikut :

Bil	Nama	Skrip	Tandatangan
1			
2	MUHAMMAD AMIR SYAFIQ BIN ZAINUDDIN		
3	MUHAMMAD WAFIUDDIN BIN CHE ABDUL HAMID		
4			

Kami telah menyerahkan skrip pada …… kepada pensyarah kelas Prof/Prof. Madya/Dr./En./Pn. ……

Tanda tangan Ketua Kumpulan

……

Tandatangan Pensyarah

……

SKRIP DIALOG DI BANDAR

:أمير مَرْحَبًا يَا أَصْدِقَائِي! هَلْ أَنتُمْ مِسْتَعِدُّونَ لِاسْتِكْشَافِ الْمَدِينَةِ الْيَوْمَ؟ Hai, kawan-kawan! Adakah anda bersedia untuk menjelajah bandar hari ini?
:إزمي .نَعَم، سَمِعْتُ أَنَّ هُنَاكَ فَعَالِيَّاتٍ فِي السَّاحَةِ الْكُبْرَى Ya, saya dengar ada acara di dataran besar.
:نصر الله صَحِيحٌ! بَعْدَ ذَٰلِكَ، هَلْ يُمْكِنُنَا الذَّهَابُ لِتَنَاوُلِ الطَّعَامِ؟ Betul! Selepas itu, boleh kita pergi makan?
:وافي مَاذَا عَنْ تَجْرِبَةِ الْأَطْعِمَةِ الْمَحَلِّيَّةِ مِثْلَ الْكَبَابِ؟ Bagaimana kalau kita mencuba masakan tempatan seperti kebab?
:أمير حَسَنًا! لِنَذْهَبْ إِلَى السَّاحَةِ. أَيَّ طَرِيقٍ سَنَأْخُذُ؟ Baiklah! Mari kita pergi ke dataran. Jalan mana yang kita ambil?
:إزمي .اتَّجِهُوا لِلْيَمِينِ، إِنَّهُ أَسْرَعُ طَرِيقٍ Ikut ke kanan, itu jalan tercepat
:نصر الله لَا نَنْسَ زِيَارَةَ الْمُتْحَفِ، يَغْلِقُ السَّاعَةَ السَّادِسَةَ Oh, dan jangan lupa tentang muzium. Ia tutup jam enam!
:وافي !وَبَعْدَ ذَٰلِكَ، سَنَذْهَبُ إِلَى السُّوقِ لِشِرَاءِ الْهَدَايَا

Dan selepas ini, kita ke pasar untuk cenderamata!
أمير:
فِكْرَةٌ رَائِعَةٌ! أُرِيدُ شِرَاءَ شَيْءٍ مُمَيَّزٍ.
Idea yang baik! Saya ingin membeli sesuatu yang khas.
إزمي:
لَا أَسْتَطِيعُ الْإِنْتِظَارَ لِرُؤْيَةِ الأَضْوَاءِ الْجَمِيلَةِ فِي الْمَسَاءِ
Saya tidak sabar untuk melihat semua lampu indah bila malam tiba!
نصر الله:
لْنَذْهَبْ بِسُرْعَةٍ! الْمَدِينَةُ بِـ انْتِظَارِنَا
Mari kita pergi dengan cepat! Bandar menanti kita!
وافي:
نَعَم، دَعُونَا نَبْدَأُ رِحْلَتَنَا
Ya, mari kita mulakan perjalanan kita!

Situasi: Kampung

Wafi:

أَيْنَ نَحْنُ الْآنَ؟

(Di mana kita sekarang?)

Amir:

.نَحْنُ فِي قَرْيَتِنَا الْجَمِيلَةِ

(Kita berada di kampung kita yang indah.)

Izmy:

.الْقَرْيَةُ هَادِئَةٌ وَجَمِيلَةٌ جِدًّا

(Kampung ini sangat tenang dan cantik.)

Wafi:

أَتُحِبُّ الزِّرَاعَ وَالْحُقُولَ هُنَا؟

(Saya suka tanam dan ladang di sini.)

Amir:

.نَعَمْ، الْهَوَاءُ لَطِيفٌ وَمُنْعِشٌ

(Ya, udara segar dan menyegarkan.)

Nasrullah:

هَلْ تَذْهَبُ إِلَى النَّهْرِ الْيَوْمَ؟

(Bolehkah kita pergi ke sungai hari ini?)

Wafi:

.فِكْرَةٌ جَيِّدَةٌ، أَنَا أُحِبُّ السِّبَاحَةَ

(Idea yang bagus, saya suka berenang.)

Amir:

.السَّمَاءُ زَرْقَاءُ وَالْجَوُّ جَمِيلٌ

(Langit biru dan cuaca indah.)

Izmy:

.الْقَرْيَةُ مَكَانٌ لِلْسَّلاَمِ وَالرَّاحَةِ

(Kampung adalah tempat yang damai dan nyaman.)

Nasrullah:

هَلْ أَنْتَ تَأْكُلُ فَاكِهَةً مِنَ الْمَزْرَعَةِ؟

(Adakah kamu makan buah dari ladang?)

Wafi:

.نَعَمْ، طَعْمُهَا لَذِيذٌ وَطَبِيعِيٌّ

(Ya, rasanya enak dan asli.)

Amir:

.الْقَرْيَةُ مَلِيئَةٌ بِالْحَيَوَانَاتِ أَيْضًا

(Kampung ini juga penuh dengan haiwan.)

Izmy:

أَتُحِبُّ أَنْ ترَى الْبَقَرَ فِي الْمَزْرَعَةِ؟

(Saya suka melihat lembu di ladang.)

Nasrullah:

.أَنَا أُحِبُّ أَنْ أَسْمَعَ صَوْتَ الطُّيُورِ هُنَا

(Saya pula suka bunyi burung di sini.)

Wafi:

تَعَالَوْا لِنَكْتَشِفَ الْمَزِيدَ.

(Mari kita datang esok dan teroka lagi.)

Nasrullah:

هَلْ تعْرِفُونَ أَيْنَ يُمْكِنُنَا الصَّيْدَ؟

(Adakah kamu tahu di mana kita boleh memancing?)

Amir:

نَعَمْ، هُنَاكَ بُحَيْرَةٌ قَرِيبَةٌ.

(Ya, ada tasik yang dekat.)

Izmy:

أُرِيدُ أَنْ أَتَعَلَّمَ كَيْفِيَّةَ الصَّيْدِ.

(Saya ingin belajar cara memancing.)

Wafi:

سَيَكُونُ مِنَ الْمُمْتِعِ أَنْ نَذْهَبَ مَعًا.

(Akan seronok jika kita pergi bersama.)

Amir:

دَعُونَا نَخْطِطَ لِذَلِكَ غَدًا.

(Mari kita rancang untuk itu esok.)

SKRIP DIALOG DI KEBUN BUAH

أمير : وَاه، هَذِهِ ٱلْمَانْجُوُ تَبْدُواْ لَذِيذَةً Wah, buah mangga ni nampak sedap!
إزمي : .بِٱلْفِعْلِ، ٱنظُرْ إِلَىٰ لَوْنِهَا، لَا بُدَّ أَنَّهَا حُلْوَةٌ Memang pun, tengok warna dia, mesti manis.
وافِي : هَيَا، مَن يُرِيدُ تَذَوُّقَ ٱلْمَانْجُوُ ٱلطَّازَجَةَ؟ Nah, siapa nak rasa mangga segar?
نصرالله : أَنَآ أُوَلَىٰ! إِذَا كَانَتْ حُلْوَةً، سَأَأْخُذُ ٱلْمَزِيدَ Aku dulu! Kalau manis, aku ambil lagi.
أمير : !لَا تَكُنْ جَشِعًا، نَصْرُ ٱللَّهِ. ٱلْجَمِيعُ يُرِيدُ التَّذَوُّقَ أَيْضًا Jangan tamak, Nashrullah. Semua nak rasa juga!
إزمي : .صَحِيحٌ، شَارِكْ ٱلْجَمِيعَ Betul tu. Kongsi-kongsi lah.
وافِي : إِنَّهَا حُلْوَةٌ جِدًّا! يَجِبُ أَنْ نَأْتِيَ هُنَا مَرَّةً أُخْرَىٰ Manis betul! Kena datang lagi nanti.
نصرالله : أَتَّفِقُ! يَجِبُ أَنْ نَجْلِبَ صَلْصَةَ ٱلْكِيشِبِ وَٱلْفِلْفِلِ ٱلْمَرَّةَ ٱلْقَادِمَةَ

Setuju! Kena bawa kicap cili pula

Transaction Management

Concurrency control

Objectives

- At the end of this lesson, you should be able to:
  - ✓ Explain **transaction support**, **properties of transaction** and the **database architecture**.
  - ✓ Explain **concurrency control**, **lost update**, **uncommitted dependency** and **inconsistent analysis problems**.
  - ✓ Explain the **serializability schedule**, **serial** and **non-serial schedule**, **non-conflict serializability precedence graph**, and its recoverability.
  - ✓ Describe the **locking**, **shared and exclusive lock**, **2PL's shrinking and growing phase**, **concurrency problems' solution by using 2PL**, **deadlock prevention** and **detection**, and **time-stamping** concept.

Transaction

- **Action**, or **series of actions**, carried out by user or application, which reads or updates contents of database.

A transaction is a **logical unit of work** on the database. May consists of:

- Entire program
- A part of a program
- A single statement (INSERT or UPDATE)
- May involve any number of operations on the database

A transaction must be entirely **completed** or **aborted**

- ✓ no **intermediate state** are acceptable
- ✓ can involve any number of reads of writes to a database.

Transaction

- Each operation on database can be considered as transactions.
- Application program is series of transactions with non-database processing in between.
- During transactions, database is transformed from consistent state to another.

Example Transaction

- Staff (staffNo, fname, lname, position, sex, DOB, salary, branchNo)
- PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

```
read(staffNo = x, salary)
salary = salary * 1.1
write(staffNo = x, new_salary)
```

- db operation: read & wrire
- Non-db operation: salary = salary * 1.1

(a)

```
delete(staffNo = x)
for all PropertyForRent records, pno
begin
    read(propertyNo = pno, staffNo)
    if (staffNo = x) then
    begin
        staffNo = newStaffNo
        write(propertyNo = pno, staffNo)
    end
end
```

(b)

a) To update the salary of a particular staff
b) To delete the member of staff with given staff no

Transaction Example

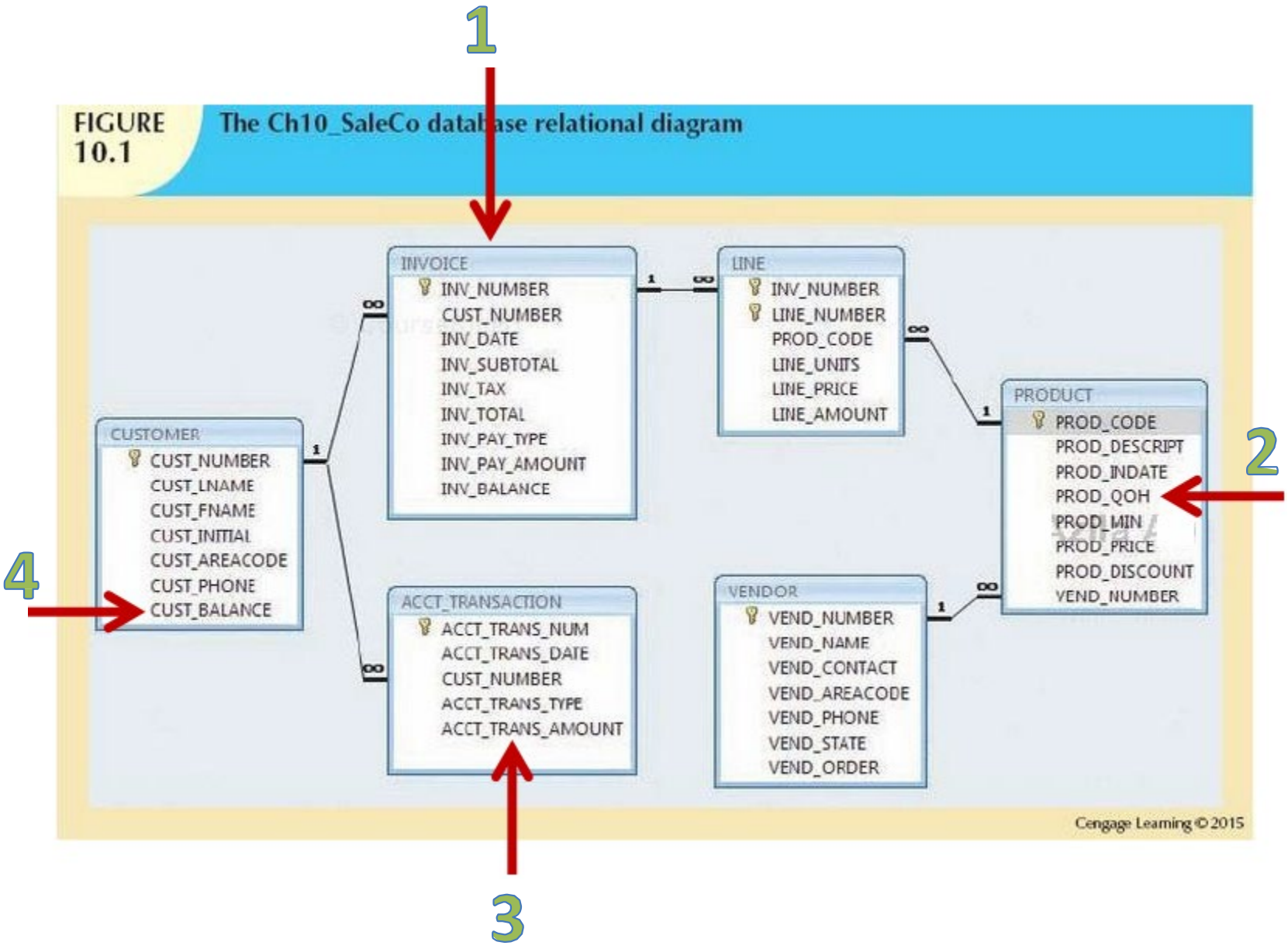
Scenario:

Sell product to a customer

Customer may charge purchase to his/her account

Action:

1. Write a new customer invoice
2. Reduce the quantity on hand in the product's inventory
3. Update the account transactions
4. Update the customer balance


1
FIGURE 10.1
The Ch10_SaleCo database relational diagram
INVOICE
INV_NUMBER
CUST_NUMBER
INV_DATE
INV_SUBTOTAL
INV_TAX
INV_TOTAL
INV_PAY_TYPE
INV_PAY_AMOUNT
INV_BALANCE
LINE
INV_NUMBER
LINE_NUMBER
PROD_CODE
LINE_UNITS
LINE_PRICE
LINE_AMOUNT
PRODUCT
PROD_CODE
PROD_DESCRIPT
PROD_INDATE
PROD_QOH
PROD_MIN
PROD_PRICE
PROD_DISCOUNT
VEND_NUMBER
2
CUSTOMER
CUST_NUMBER
CUST_LNAME
CUST_FNAME
CUST_INITIAL
CUST_AREACODE
CUST_PHONE
CUST_BALANCE
4
ACCT_TRANSACTION
ACCT_TRANS_NUM
ACCT_TRANS_DATE
CUST_NUMBER
ACCT_TRANS_TYPE
ACCT_TRANS_AMOUNT
VENDOR
VEND_NUMBER
VEND_NAME
VEND_CONTACT
VEND_AREACODE
VEND_PHONE
VEND_STATE
VEND_ORDER
3
Cengage Learning © 2015

Evaluating Transaction Result

Examine CUSTOMER table to determine current balance for customer 10016

```
SELECT CUST_NUMBER, CUST_BALANCE
FROM CUSTOMER
WHERE CUST_NUMBER = 10016
```

- Query does not make any changes to database
- Because the transaction did not alter the database, the database remains in consistent state after the access

Airline Transaction Example

START TRANSACTION
    Display greeting
    Get reservation preferences from user
    SELECT departure and return flight records
    If reservation is acceptable, then
        UPDATE seats remaining of departure flight record
        UPDATE seats remaining of return flight record
        INSERT reservation record
        Print ticket if requested
    End If
    On Error: **ROLLBACK**
COMMIT

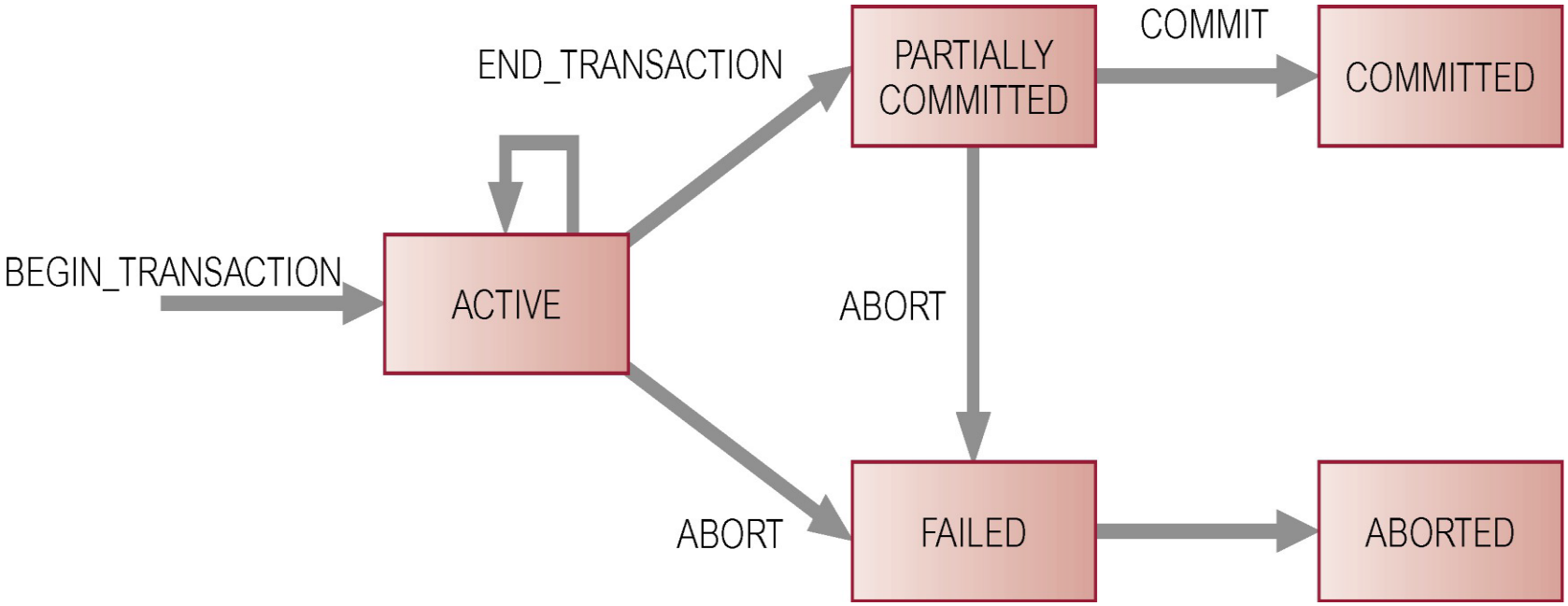
ATM Transaction Example

START TRANSACTION
    Display greeting
    Get account number, pin, type, and amount
    SELECT account number, type, and balance
    If balance is sufficient then
        UPDATE account by posting debit
        UPDATE account by posting credit
        INSERT history record
        Display message and dispense cash
        Print receipt if requested
    End If
    On Error: **ROLLBACK**
COMMIT

Transaction Support

- Can have one of two outcomes:
  - ✓ **Success** - transaction commits and database reaches a new consistent state.
  - ✓ **Failure** - transaction aborts, and database must be restored to consistent state before it started.
    - Such a transaction is rolled back or undone.
- Committed transaction cannot be aborted.
- Aborted transaction that is rolled back can be restarted later.

State Transition Diagram for Transaction



- PARTIALLY COMMITED: occurs after final statement has been executed.
  - If system fail (any data updated not safely recorded on secondary storage), the transaction would go to FAILED state & have to be aborted.
  - If the transaction successful, any updates can be recorded – transaction go to COMMITED state.
- FAILED: occurs if the transaction cannot be committed or aborted while in the ACTIVE state

Transaction Properties

Four basic *(ACID)* properties of a transaction are:

Atomicity

- '**All or nothing**' property.
- All operations of a transaction must be **completed**, if not, the transaction is **aborted**

Consistency

- Must transform database from one **consistent state** to another.

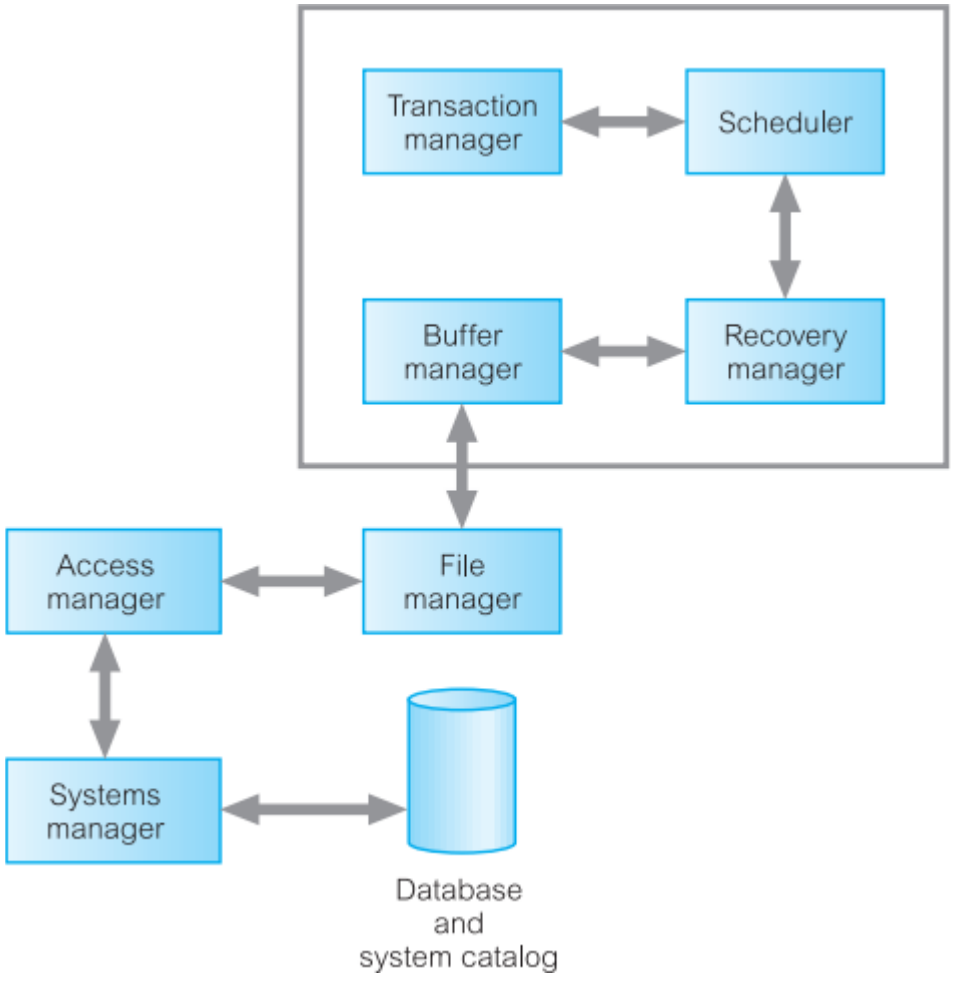
Isolation

- **Transactions** execute **independently** of **one another**. Partial effects of incomplete transactions should not be visible to other transactions.
- Data used during transaction cannot be used by second transaction until the first is completed.

Durability

- Effects of a committed transaction are **permanent** and must **not be lost because of later failure**.

DBMS Transaction Subsystem (Database architecture)



4 database modules that handles transactions, concurrency control, and recovery

Transaction manager: Coordinates transaction on behalf of application program. Communicates with Scheduler.

Scheduler: Module responsible for implementing a particular strategy for concurrency control. Sometimes referred to as the lock manager if the concurrency protocol is locking-based.

Recovery manager: Ensure database is restored to the state it was in before the start of the transaction and therefore, a consistent state.

Buffer manager: Responsible for the efficient transfer of data between disk storage and main memory.

Concurrency Control

- The process of **managing simultaneous operations** on the database without having them interfere with one another.

- What?

Coordination of the simultaneous transactions execution in a multiuser database system.

- Why? (Objective)

To ensure serializability of transactions in a multiuser database environment.

The Need for Concurrency Control

- The main objective of concurrency control is to allow many users perform different operations at the same time.
- It increases the throughput because of handling multiple transactions simultaneously.

- ✓ Transaction **can interleave** with each other, but it **cannot interfere**.
- ✓ If 2 users want to update the same bank account at the same time, there will be some incorrect balance at the end of the transactions.
- ✓ Although two transactions may be correct in themselves, interleaving of operations may produce an incorrect result.

The Need for Concurrency Control

- If there is no concurrency control, the problems that might occur are:
  - ✓ **Lost Update Problem**
  - ✓ **Uncommitted Dependency Problem**
  - ✓ **Inconsistent Analysis Problem**

Lost Update Problem

- Occurs in two concurrent transactions when:
  - Same data element is updated
  - One of the updates is lost (successfully completed update is overridden by another user)

Time	T1	T2	Bal_x
t1		Begin_transaction	100
t2	Begin_transaction	Read(bal_x)	100
t3	read (bal_x)	$Bal_x = bal_x + 100$	100
t4	$Bal_x = Bal_x - 10$	Write (bal_x)	200
t5	Write (bal_x)	Commit	90
t6	Commit		90

Loss of T2's update is avoided by preventing T1 from reading bal_x until after update

Lost Update Problem

- Occurs when two concurrent transactions, T1 and T2, are updating the same data element and one of the update is lost (**overwritten** by the other transaction).

Lost update: a concurrency control problem in which one user's update overwrites another user's update

Lost Update Problem

- T1 withdrawing 10 from an account with balx, initially 100.
- T2 depositing 100 into same account.
- Serially, final balance would be 190.
- The addition of 100 units is “lost” during the process.

Uncommitted Dependency Problem

- Occurs when:
  - Two transactions are executed concurrently
  - First transaction is rolled back after the second transaction has already accessed uncommitted data

Time	T3	T4	Bal_x
t1		Begin_transaction	100
t2		read (bal_x)	100
t3		$Bal_x = bal_x + 100$	100
t4	Begin_transaction	Write (bal_x)	200
t5	read (bal_x)		200
t6	$Bal_x = Bal_x - 10$	Rollback	100
t7	Write (bal_x)		190
t8	Commit		190

T3 and T4, are executed concurrently and T4 is rolled back after T3 has already accessed the uncommitted data - thus violating the isolation property of transactions.

Uncommitted Dependency Problem

- T4 updates bal_x to 200 but it aborts, so bal_x should be back at original value of 100.

- T3 has read new value of bal_x (200) and uses value as basis of 10 reduction, giving a new balance of 190, instead of 90.

Inconsistent Analysis Problem

- Occurs when a transaction accesses data before and after another transaction(s) finish working with such data.
  - e.g. T1 calculates some summary (aggregate) function over a set of data while another transaction (T2) was updating the same data.
  - Transaction might read some data **before** they are changed and other data **after** they are changed, thereby yielding inconsistent result

Inconsistent Analysis Problem

- Occurs when transaction reads several values but second transaction updates some of them during execution of first.

Time	T_5	T_6	bal_x	bal_y	bal_z	sum
t_1		begin_transaction	100	50	25	
t_2	begin_transaction	sum = 0	100	50	25	0
t_3	read(bal_x)	read(bal_x)	100	50	25	0
t_4	$bal_x = bal_x - 10$	sum = sum + bal_x	100	50	25	100
t_5	write(bal_x)	read(bal_y)	90	50	25	100
t_6	read(bal_z)	sum = sum + bal_y	90	50	25	150
t_7	$bal_z = bal_z + 10$		90	50	25	150
t_8	write(bal_z)		90	50	35	150
t_9	commit	read(bal_z)	90	50	35	150
t_{10}		sum = sum + bal_z	90	50	35	185
t_{11}		commit	90	50	35	185

Problem is avoided by preventing T6 from reading bal_x and bal_z until after T5 completed updates

Inconsistent Analysis Problem

- Sometimes referred to as dirty read or unrepeatable read.
- T6 is totaling balances of account x (100), account y (50), and account z (25).
- In the meantime, T5 has transferred 10 from bal_x to bal_z, so T6 now has wrong result (10 too high).

Serializability

- **Objective** of a concurrency control protocol is **to schedule transactions** in such a way as **to avoid any interference** hence prevent the concurrency problem.

- ✓ Could run transactions serially, but this limits degree of concurrency or parallelism in system.
- ✓ The aim of multi-user DBMS is to maximize the degree of concurrency or parallelism in the system, so the transactions that can execute without interfering one another can run in parallel.
- ✓ Serializability identifies those executions of transactions that are guaranteed to ensure consistency.

Serializability

Schedule	Sequence of reads/writes by set of concurrent transactions.
Serial Schedule	Schedule where operations of each **transaction are executed consecutively** without any interleaved operations from other transactions
Nonserial Schedule	Schedule where operations from **set of concurrent transactions are interleaved**.

Serializability

- How do we ensure **database** are in a **consistent state**?
  - ✓ It is when we have a serial schedule/serial in nature
- **Serializability** is a concept that helps us to **check which schedules are serializable**.
- A **serial schedule is always a serializable schedule** ➔ because in serial schedule, a transaction only starts when the other transaction finished execution.
- A serializable schedule is the one that always leaves the database in consistent state.
- If a set of transactions executes concurrently, we say that the (nonserial) schedule is correct if it **produces the same results** as some **serial execution**. This schedule is called **serializable**.

Serializability

In serializability, ordering of read/writes is important:

a) If two transactions **only read a data item**, they **do not conflict,** and order is not important.

b) If two transactions **either read or write** completely **separate data items**, they **do not conflict,** and order is not important.

c) If one transaction **writes a data item** and **another reads or writes same data item**, order of execution is important.

Time	T_7	T_8
t_1	begin_transaction	
t_2	read(bal_x)	
t_3	write(bal_x)	
t_4		begin_transaction
t_5		read(bal_x)
t_6		write(bal_x)
t_7	read(bal_y)	
t_8	write(bal_y)	
t_9	commit	
t_{10}		read(bal_y)
t_{11}		write(bal_y)
t_{12}		commit

(a)

T_7	T_8
begin_transaction	
read(bal_x)	
write(bal_x)	
	begin_transaction
	read(bal_x)
read(bal_y)	
	write(bal_x)
write(bal_y)	
commit	
	read(bal_y)
	write(bal_y)
	commit

(b)

T_7	T_8
begin_transaction	
read(bal_x)	
write(bal_x)	
read(bal_y)	
write(bal_y)	
commit	
	begin_transaction
	read(bal_x)
	write(bal_x)
	read(bal_y)
	write(bal_y)
	commit

(c)

Equivalent schedule:
(a) Nonserial Schedule S1;
(b) Nonserial Schedule S2;
(c) Serial Schedule S3, equivalent to S1 and S2

This type of serializability is known as conflict serializability.
A conflict serializable schedule orders any conflicting operations in same way as some serial execution.

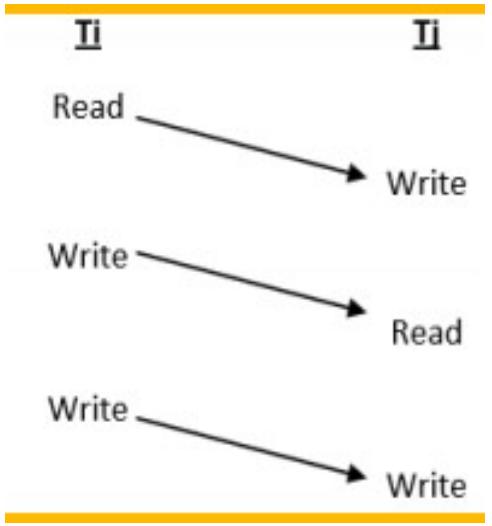
Conflict Serializable

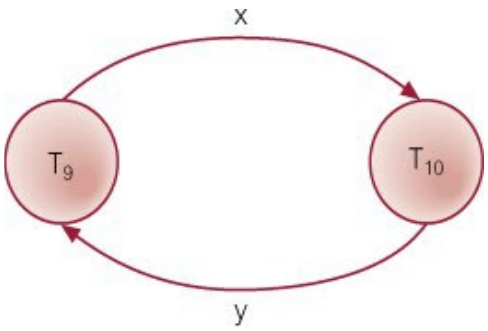
- A schedule is called conflict serializable if we can convert it into a serial schedule after swapping its non-conflicting operations.

Testing for Conflict Serializibility

- Use precedence graph to test for conflict serializability.

1. Create a node for each transaction (N)
2. Create directed edges (E)
   - ✓ a directed edge Ti → Tj, if Tj reads the value of an item written by Ti;
   - ✓ a directed edge Ti → Tj, if Tj writes a value into an item after it has been read by Ti.
   - ✓ a directed edge Ti → Tj, if Tj writes a value into an item after it has been written by Ti.





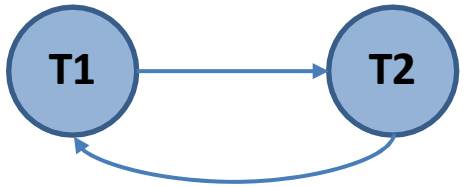
- If precedence graph contains cycle schedule is not conflict serializable.

Serializability

S1	
T1	T2
Read A	
Write A	
	Read A
	Write A

S2	
T1	T2
Read A	
	Read A
Write A	
	Write A

No cycle ➔ **serial graph**



Has cycle ➔ **parallel (nonserial) schedule**

Can we change from parallel to serial?

Why parallel is not ok? ➔ cannot ensure the database consistent state

Example on Precedence Graph

S = [$R_1(Z)$, $R_2(Y)$, $W_2(Y)$, $R_3(Y)$, $R_1(X)$, $W_1(X)$, $W_1(Z)$, $W_3(Y)$, $R_2(X)$, $R_1(Y)$, $W_1(Y)$, $W_2(X)$, $R_3(W)$, $W_3(W)$]

- By using precedence graph, determine whether this transaction is serializable or not.

Testing for Conflict Serializability

https://youtu.be/odik_Bbg5Lk

Concurrency Control Techniques

- Two basic concurrency control techniques:
  - ✓ Locking
  - ✓ Timestamping
- Both are conservative approaches: delay transactions in case they conflict with other transactions.
- Optimistic methods assume conflict is rare and only check for conflicts at commit.

Locking

- Transaction uses **locks to deny access to other transactions** and to **prevent incorrect updates**.
- Most widely used approach to ensure serializability.
- Generally, a transaction must claim a **shared (read)** or **exclusive (write) lock** on a data item before read or write.
- Lock prevents another transaction from modifying item or even reading it, in the case of a write lock.

Locking

- If transaction has **shared lock (S)** on item, it **can read but not update** it.
- If transaction has **exclusive lock (X)** on item, **can both read and update** item.
- **Reads cannot conflict**, so more than one transaction can hold shared locks (S) simultaneously on same item.
- Exclusive lock (x) gives transaction **exclusive access** to that item.
- Some systems **allow transaction** to upgrade read lock to an exclusive lock, or downgrade exclusive lock to a shared lock
- **Lock manager**: Responsible for assigning and policing the locks used by the transactions

Locking Conflicts

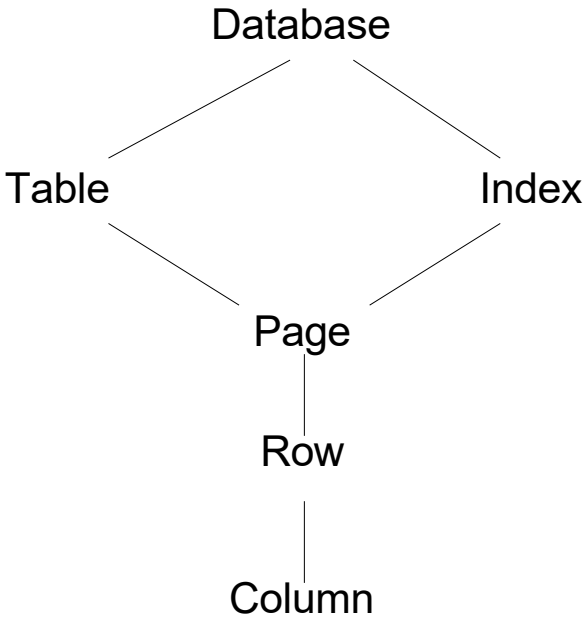
	USER 2 REQUESTS	
USER 1 HOLDS	S Lock	X Lock
S Lock	Lock Granted	User 2 Waits
X Lock	User 2 Waits	User 2 Waits

- A shared (S) lock must be obtained before reading a database item, whereas an exclusive (X) lock must be obtained before writing
- Any number of users can hold a S lock on the same item but only one user can hold X lock

Lock Granularity

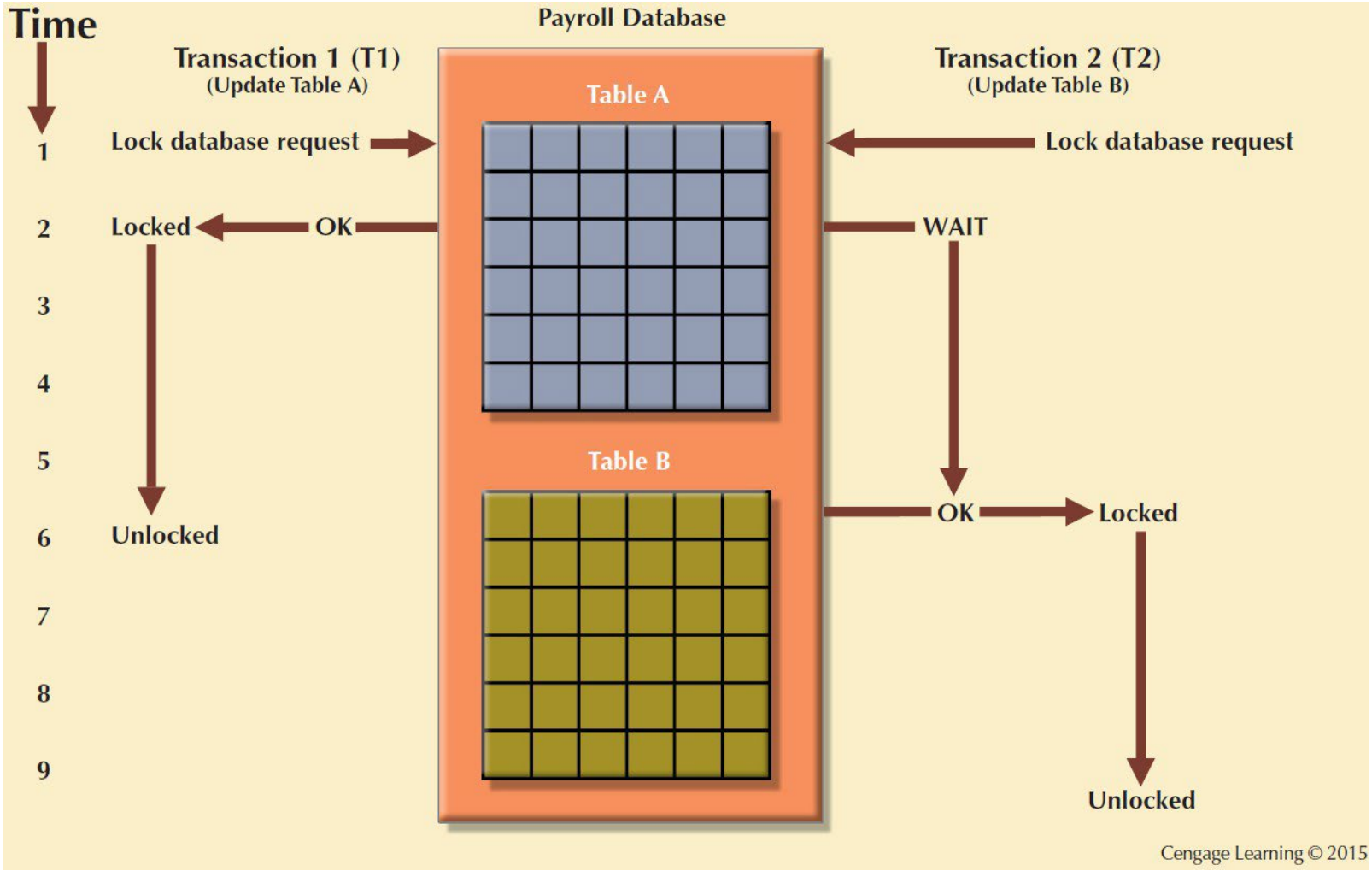
- Indicates the level of lock use
- Levels of locking
  - **Database-level lock**
  - **Table-level lock**
  - **Page-level lock**
    - **Page** or **diskpage**: Directly addressable section of a disk
  - **Row-level lock**
  - **Field-level lock**

Lock Granularity

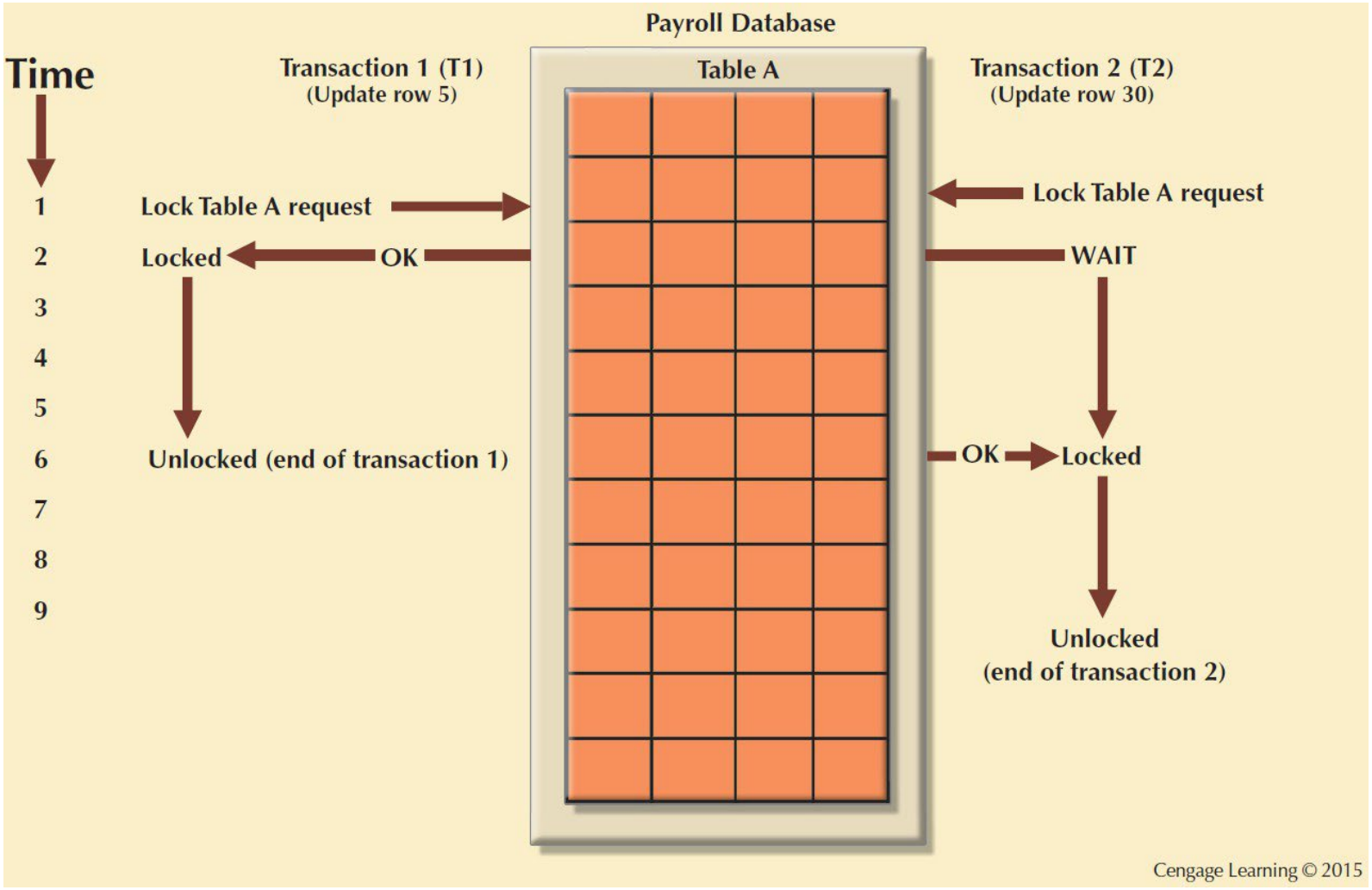


- The size of the database item locked
- A trade-off between waiting time (amount of concurrency permitted) and overhead (number of locks held)

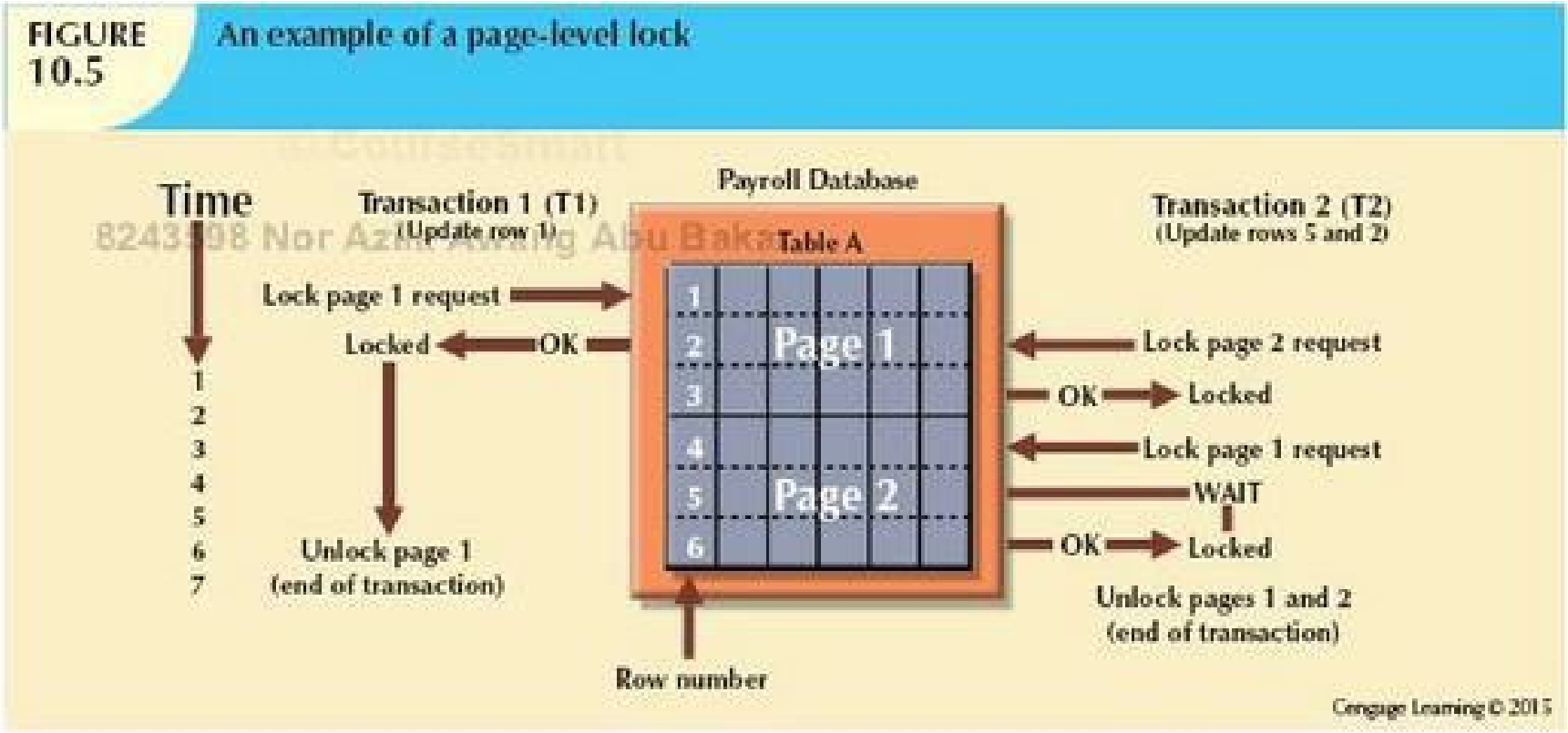
Database-Level Locking Sequence



An Example of a Table-Level Lock



An Example of a Page-Level Lock



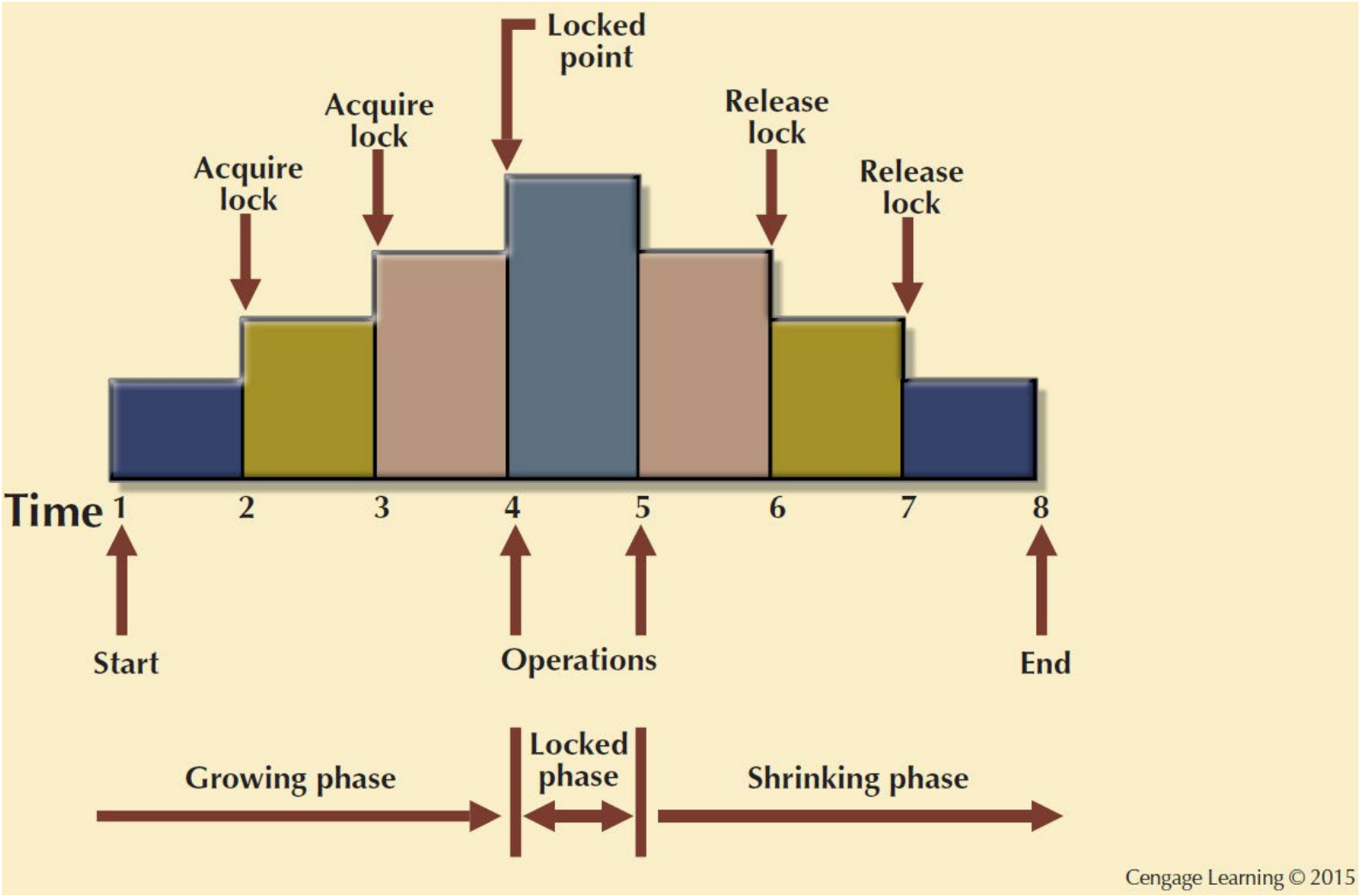
Two-Phase Locking (2PL)

- Defines how transactions **acquire** and **relinquish locks**.
- Guarantees serializability but does not prevent deadlocks.
- Two phases for transaction:
  1. **Growing phase** – Transaction acquires all locks but cannot release any locks.
  2. **Shrinking phase** – Transaction releases all locks but cannot acquire any new locks.

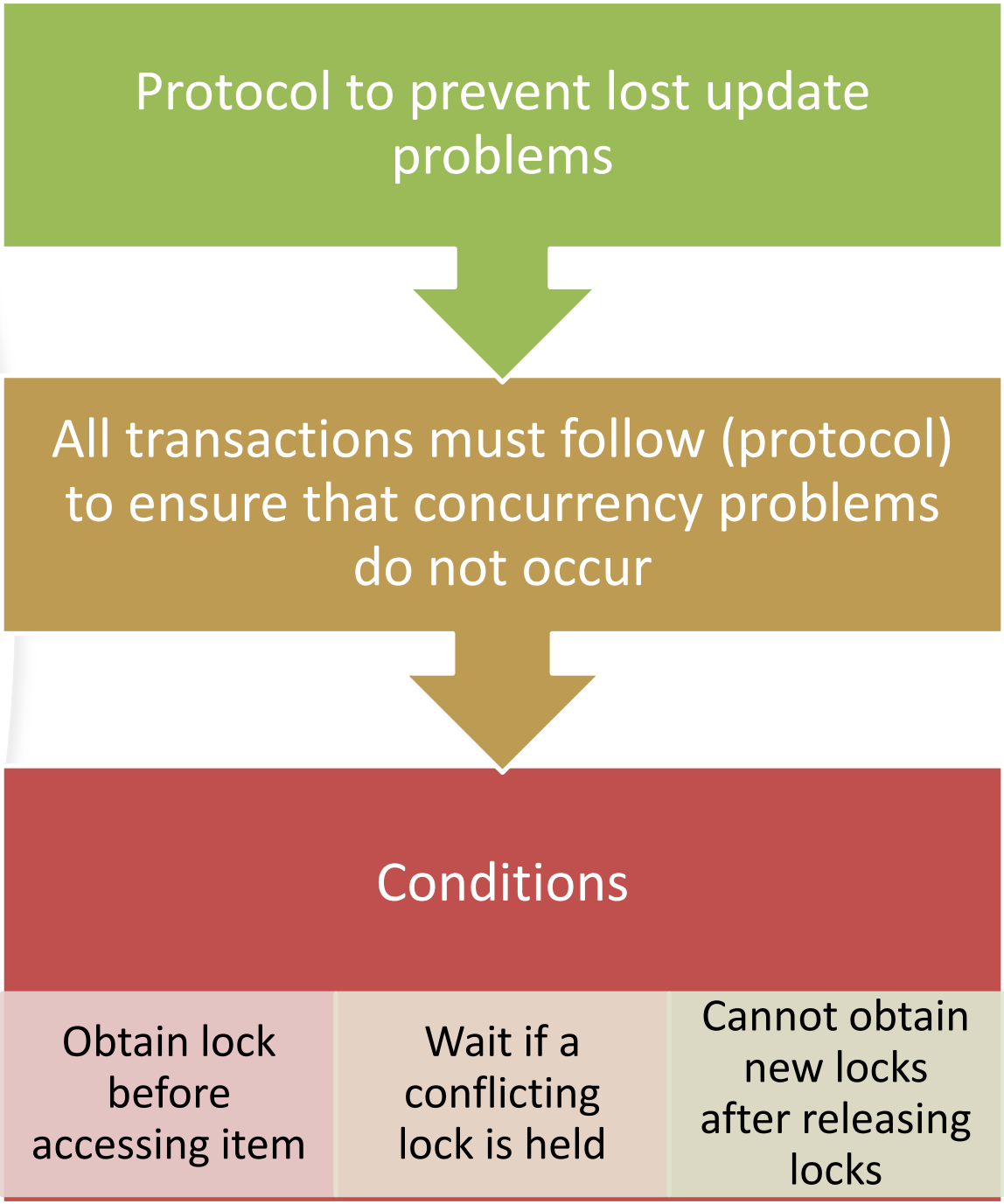
Two-Phase Locking (2PL)

- Governing rules
  - Two transactions cannot have conflicting locks
  - No unlock operation can precede a lock operation in the same transaction
  - No data are affected until all locks are obtained

Two-Phase Locking Protocol



Two Phase Locking (2PL) Protocol



Preventing Lost Update Problem Using 2PL

Time	T1	T2	Bal_x
t1		Begin_transaction	100
t2	Begin_transaction	write_lock (bal_x)	100
t3	Write_lock (bal_x)	read (bal_x)	100
t4	WAIT	$Bal_x = bal_x + 100$	100
t5	WAIT	Write (bal_x)	200
t6	WAIT	Commit/unlock(bal_x)	200
t7	read (bal_x)		200
t8	$Bal_x = Bal_x - 10$		200
t9	Write (bal_x)		190
t10	Commit/unlock (bal_x)		190

Preventing Uncommited Dependency Problem Using 2PL

Time	T3	T4	Bal_x
t1		Begin_transaction	100
t2		write_lock (bal_x)	100
t3		read (bal_x)	100
t4	Begin_transaction	$Bal_x = bal_x + 100$	100
t5	Write_lock (bal_x)	Write (bal_x)	200
t6	WAIT	Rollback/unlock (bal_x)	100
t7	read (bal_x)		100
t8	$Bal_x = Bal_x - 10$		100
t9	Write (bal_x)		90
t10	Commit/unlock (bal_x)		90

Preventing Inconsistent Retrievals Problem using 2PL

Time	T_5	T_6	bal_x	bal_y	bal_z	sum
t_1		begin_transaction	100	50	25	
t_2	begin_transaction	sum = 0	100	50	25	0
t_3	write_lock(bal_x)		100	50	25	0
t_4	read(bal_x)	read_lock(bal_x)	100	50	25	0
t_5	$bal_x = bal_x - 10$	WAIT	100	50	25	0
t_6	write(bal_x)	WAIT	90	50	25	0
t_7	write_lock(bal_z)	WAIT	90	50	25	0
t_8	read(bal_z)	WAIT	90	50	25	0
t_9	$bal_z = bal_z + 10$	WAIT	90	50	25	0
t_{10}	write(bal_z)	WAIT	90	50	35	0
t_{11}	commit/unlock(bal_x, bal_z)	WAIT	90	50	35	0
t_{12}		read(bal_x)	90	50	35	0
t_{13}		sum = sum + bal_x	90	50	35	90
t_{14}		read_lock(bal_y)	90	50	35	90
t_{15}		read(bal_y)	90	50	35	90
t_{16}		sum = sum + bal_y	90	50	35	140
t_{17}		read_lock(bal_z)	90	50	35	140
t_{18}		read(bal_z)	90	50	35	140
t_{19}		sum = sum + bal_z	90	50	35	175
t_{20}		commit/unlock(bal_x, bal_y, bal_z)	90	50	35	175

Deadlocks

- An impasse that may result when two (or more) transactions are each **waiting for locks** held by the other to be released.
- Only one way to break deadlock: abort one or more of the transactions.
- Deadlock should be transparent to user, so DBMS should restart transaction(s).

How a Deadlock Condition is Created

Time	T_{17}	T_{18}
t_1	begin_transaction	
t_2	write_lock($\mathbf{bal_x}$)	begin_transaction
t_3	read($\mathbf{bal_x}$)	write_lock($\mathbf{bal_y}$)
t_4	$\mathbf{bal_x} = \mathbf{bal_x} - 10$	read($\mathbf{bal_y}$)
t_5	write($\mathbf{bal_x}$)	$\mathbf{bal_y} = \mathbf{bal_y} + 100$
t_6	write_lock($\mathbf{bal_y}$)	write($\mathbf{bal_y}$)
t_7	WAIT	write_lock($\mathbf{bal_x}$)
t_8	WAIT	WAIT
t_9	WAIT	WAIT
t_{10}	⋮	WAIT
t_{11}	⋮	⋮

Deadlocks - Control techniques

- Timeouts
- Deadlock prevention
- Deadlock detection and recovery

Deadlock - Timeouts

- **Transaction** that **requests lock** will only **wait for a system defined period**.
- If lock has **not been granted** within this period, **lock request times out**.
- In this case, **DBMS assumes transaction may be deadlocked**, even though it may not be, and it aborts and automatically restarts the transaction.

Deadlock - Prevention

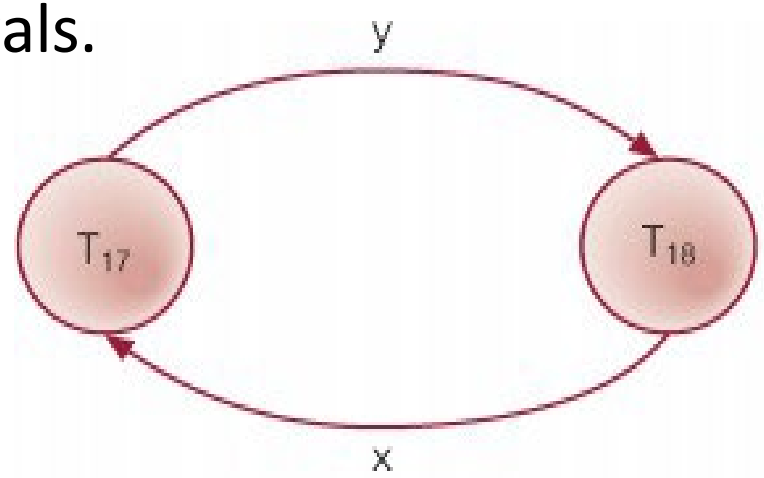
- DBMS looks ahead to see if transaction would cause deadlock and never allows deadlock to occur.
- Could order transactions using transaction timestamps.

Wait-Die	only an older transaction can wait for younger one, otherwise transaction is aborted (dies) and restarted with same timestamp
Wound-Wait	only a younger transaction can wait for an older one. If older transaction requests lock held by younger one, younger one is aborted (wounded).

https://youtu.be/WZtebOyiu0M

Deadlock Detection and Recovery

- DBMS allows deadlock to occur but recognizes it and breaks it.
- Usually handled by construction of wait-for graph (WFG) showing transaction dependencies:
  - Create a node for each transaction.
  - Create edge Ti ➔ Tj, if Ti waiting to lock item locked by Tj.
- Deadlock exists if and only if WFG contains cycle.
- WFG is created at regular intervals.



Timestamping Methods

Timestamp:

A unique identifier created by the DBMS that indicates the relative starting time of a transaction.

Timestamping:

A concurrency control protocol that orders transactions in such a way that older transactions, transactions with *smaller* timestamps, get the priority in the event of conflict.

Exercise 1

a) Consider the following schedule:

S = [R3(Z), R2(X), W1(X), W3(Y), R1(Y), W2(Z), W1(Z)]

i. Draw the transaction table and by referring above schedule
ii. Draw a precedence or a serialization graph for the schedule S.
iii. Is the schedule serializable?

Exercise 2

Check whether a schedule is conflict serializable or not. Produce a precedence graph.

T1	T2	T3
R(x)		
		R(y)
		R(x)
	R(y)	
	R(z)	
		W(y)
	W(z)	
R(z)		
W(x)		
W(z)		

https://youtu.be/nvXWxObRR1U

Exercise 3

Consider the transactions T1 and T2 below:

T1	Time	T2	Value
Read Balance	1		1200
Balance = Balance + 1000	2		1200
Write Balance	3	Read Balance	2200
Rollback	4	Balance = Balance – 1000	1200
	5	Write Balance	1200

i) Name the interference problem above.

(1 mark)

ii) Rewrite the transactions so that they obey the two-phase locking protocol

(6 marks)

Exercise 4

Given the following transactions T1 and T2:

Time	T1	T2	A	B
1		Begin Trans	200	150
2	Begin Trans	Read (A)		
3	Read (A)	A=A-X		
4	Read (B)	Read (B)		
5	If (A>B) then A=A+X	B=B+Y		
6		Write (A)		
7	Write (A)	Write (B)		
8	Commit	Commit		

a) Name the concurrency problem for the transaction above. (1 mark)

b) Suppose X=80 and Y=65, fill the details for A and B of the above transactions. (6 marks)

c) What is the final value of A and B if T1 and T2 are executed serially? (4 marks)

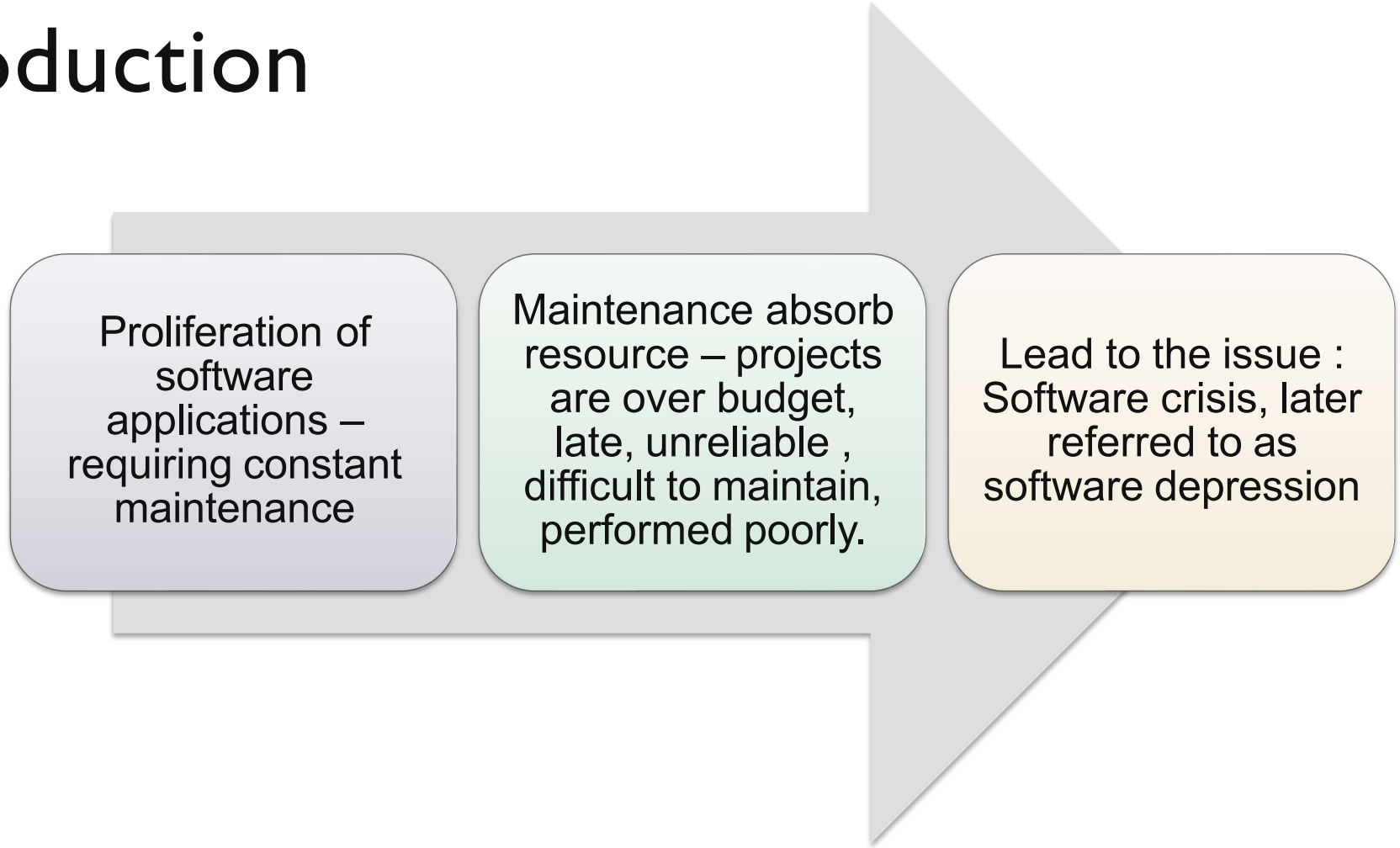
d) Rewrite the above transactions so that they obey the two-phased locking (2PL). (6 marks)

DATABASE SYSTEM DEVELOPMENT LIFECYCLE

Objectives

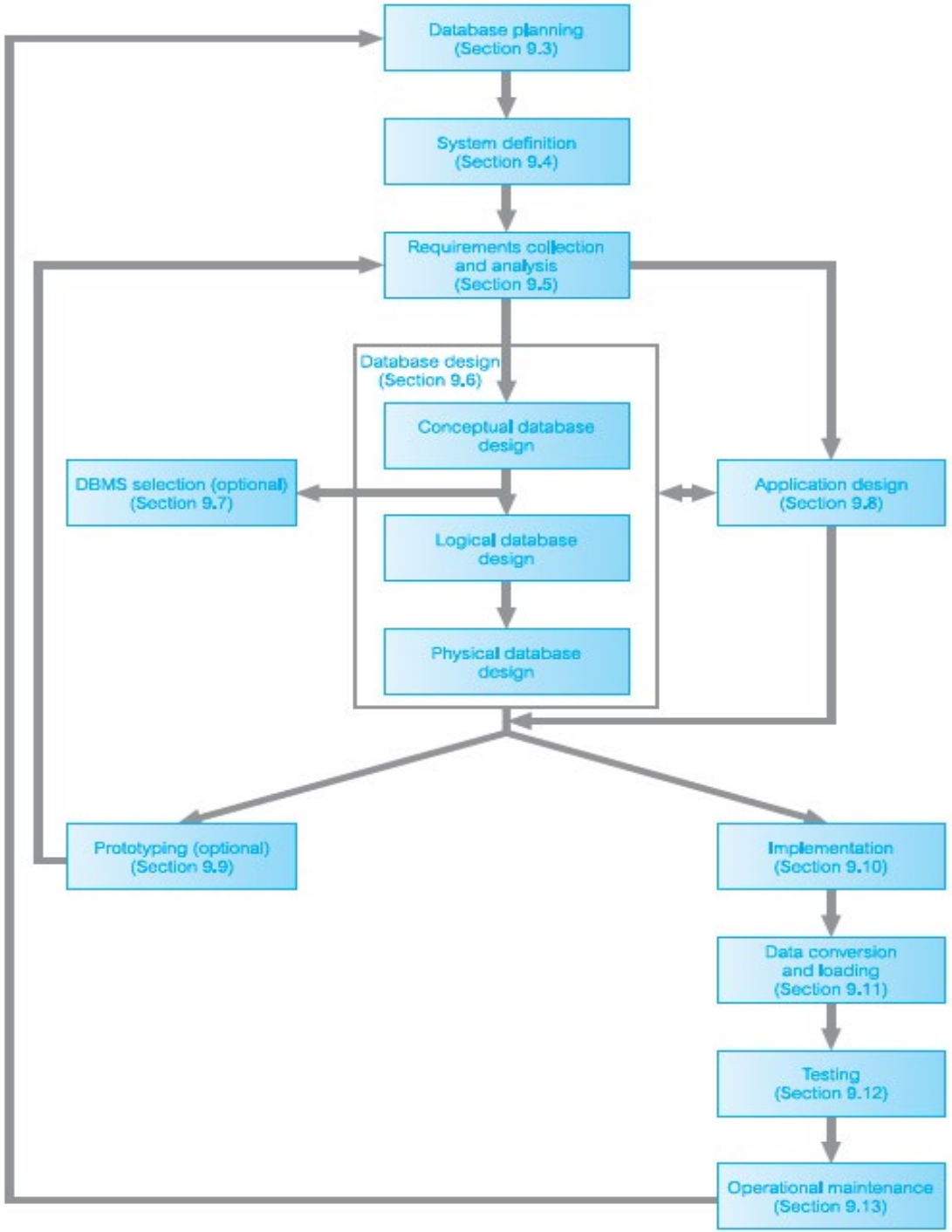
- Describe database development lifecycle
- Explain information system lifecycle
- Explain the steps in conceptual database design
- Explain the steps in logical database design
- Explain the steps in physical database design

Introduction



Information Systems Lifecycle

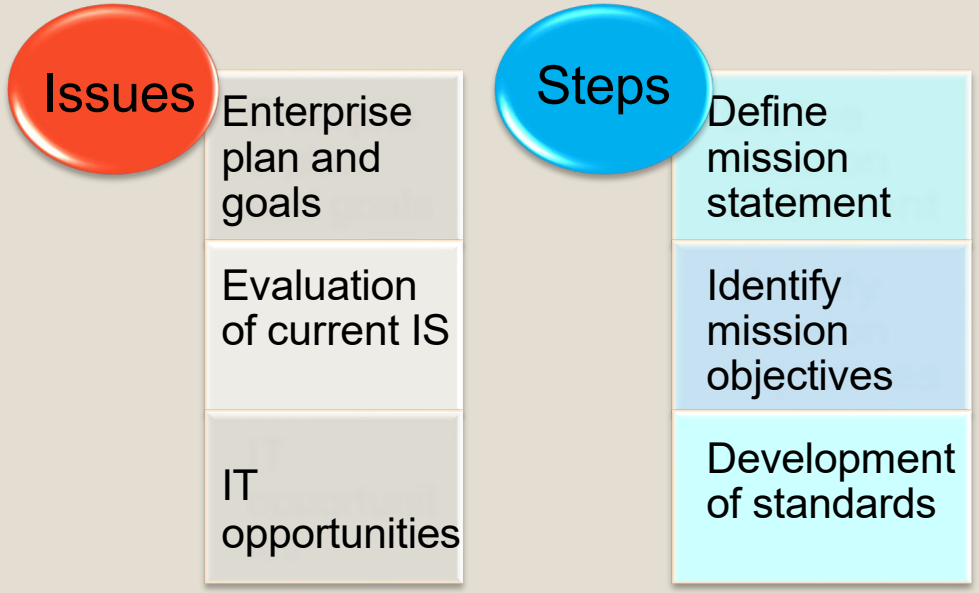
- Information system (IS) : The resources that enable the collection, management, control, and dissemination of information throughout an organization.
- Database is a fundamental component of an IS.
- Lifecycle of an organization's IS is inherently linked to the lifecycle of the database system that supports it.



Database System Development Lifecycle

Database Planning

- Database Planning: The management activities that allow stages of database system development lifecycle to be realized as efficiently and effectively as possible.



System Definition

- System Definition: Describe the scope and boundaries of the database system and the major user views

- User View: Define what is required of a database system from perspective of:
  - a particular job role (such as Manager or Supervisor) or
  - enterprise application area (such as marketing, personnel, or stock control).

Requirements Collection and Analysis

- Requirements Collection and Analysis: Process of collecting and analyzing information about the part of organization to be supported by the database system and using this information to identify users' requirements of new system.

- Information is gathered for each major user view including:
  - a description of data used or generated;
  - details of how data is to be used/generated;
  - any additional requirements for new database system

Database Design

Database Design: Process of creating a design for a database that will support the enterprise's mission statement and mission objectives for the required database system.

Design Methodology

Design Methodology: A structured approach that uses procedures, techniques, tools, and documentation aids to support and facilitate the process of design.

Critical Success Factors in Database Design

Work interactively with the users as much as possible.	Follow a structured methodology throughout the data modeling process.	Employ a data-driven approach.
Incorporate structural and integrity considerations into the data models.	Combine conceptualization, normalization, and transaction validation techniques into the data modeling methodology.	Use diagrams to represent as much of the data models as possible.
Use a Database Design Language (DBDL) to represent additional data semantics.	Build a data dictionary to supplement the data model diagrams.	Be willing to repeat steps.

Database Design

Approaches

- Bottom-Up
- Top-down
- Inside-out
- Mixed Strategy

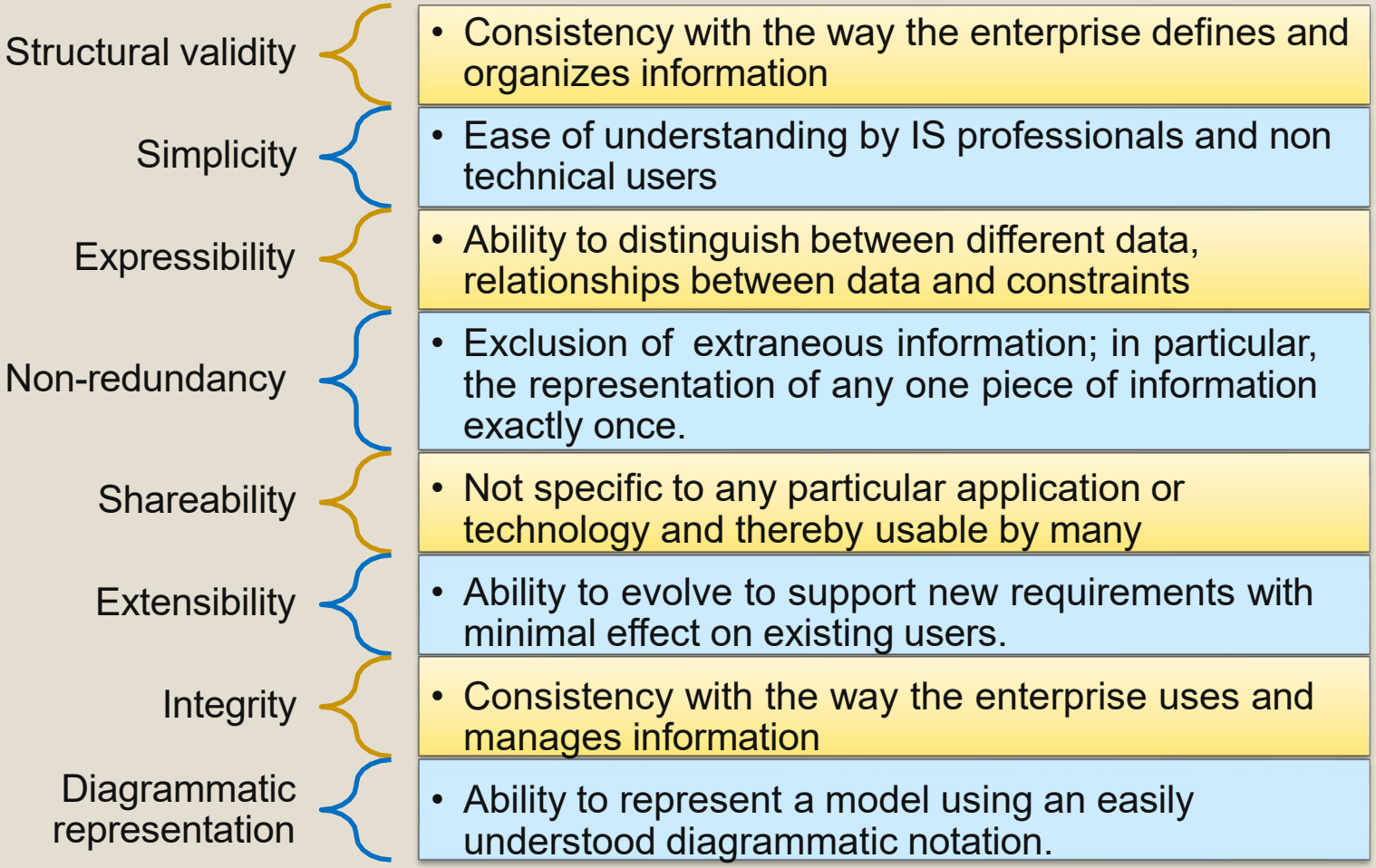
Data Modeling

- to assist in understanding the meaning (semantics) of the data;
- to facilitate communication about the information requirements.

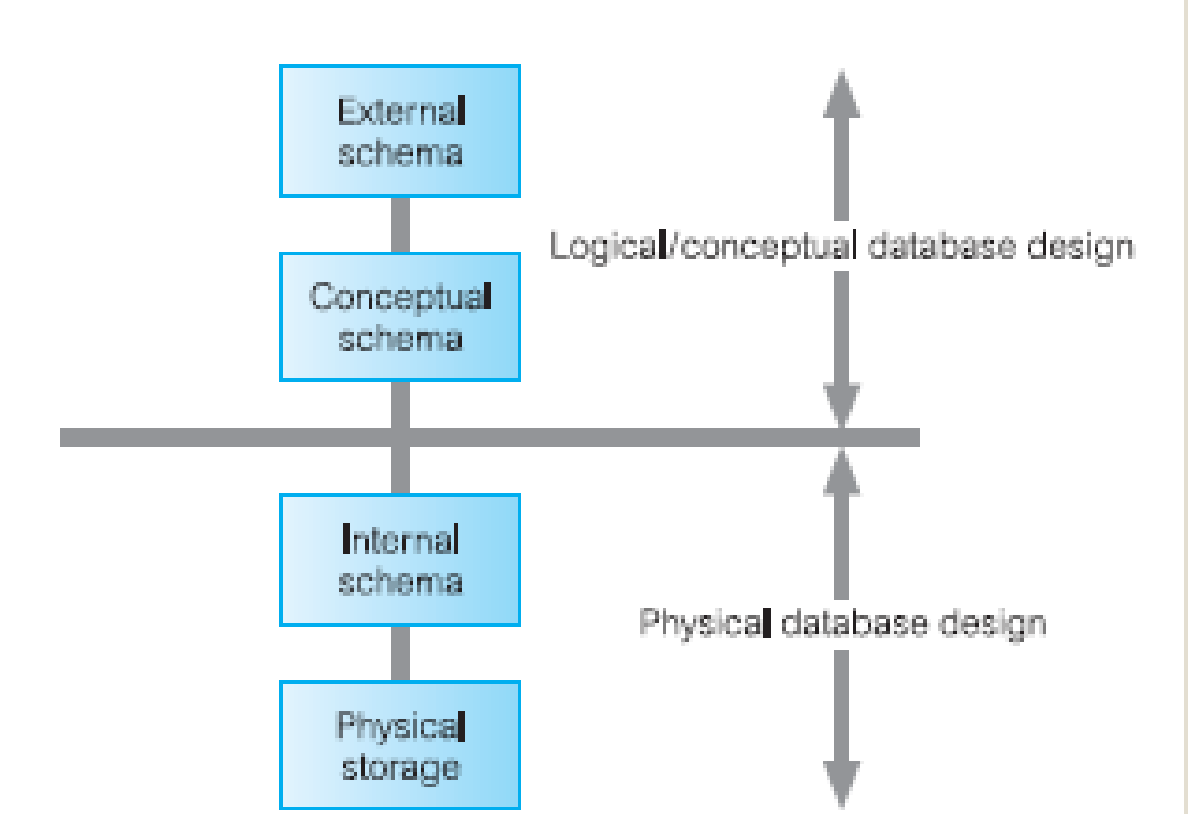
Phase of Database Design

- Conceptual database design
- Logical database design
- Physical database design

Criteria to produce optimal data model



3 Level of ANSI-SPARC Architecture and Phases of Database Design



DBMS Selection

- DBMS Selection: Selection of an appropriate DBMS to support the database system.
- Undertaken at any time prior to logical design provided sufficient information is available regarding system requirements.
- Main steps to selecting a DBMS:
  - define Terms of Reference of study;
  - shortlist two or three products;
  - evaluate products;
  - recommend selection and produce report.

Application Design

- Application Design: Design of user interface and application programs that use and process the database.
- Database design and application design are parallel activities.
- Includes two important activities:
  - transaction design (retrieval, update, mixed transaction);
  - user interface design.

Prototyping

- Building working model of a database system

Implementation

- Physical realization of the database and application designs.

Data Conversion and Loading

- Transferring any existing data into new database and converting any existing applications to run on new database.

Testing

- Process of running the database system with intent of finding errors.

Operational Maintenance

- Process of monitoring and maintaining database system following installation.

“

STEPS IN DATABASE DESIGN

STEP 1: Conceptual Database Design

- The process of constructing a model of the data used in an enterprise, independent of ***all*** physical considerations
- Obj: To build a conceptual data model of the data requirements of the enterprise.
  - Model comprises
  - entity types
  - relationship types
  - attributes and attribute domains
  - primary and alternate keys
  - integrity constraints.

Step 1.1 Identify entity types

Step 1.2 Identify relationship types

Step 1.3 Identify and associate attributes with entity or relationship types

Step 1.4 Determine attribute domains

Step 1.5 Determine candidate, primary, and alternate key attributes

Step 1.6 Consider use of enhanced modeling concepts (optional step)

Step 1.7 Check model for redundancy

Step 1.8 Validate conceptual model against user transactions

Step 1.9 Review conceptual data model with user

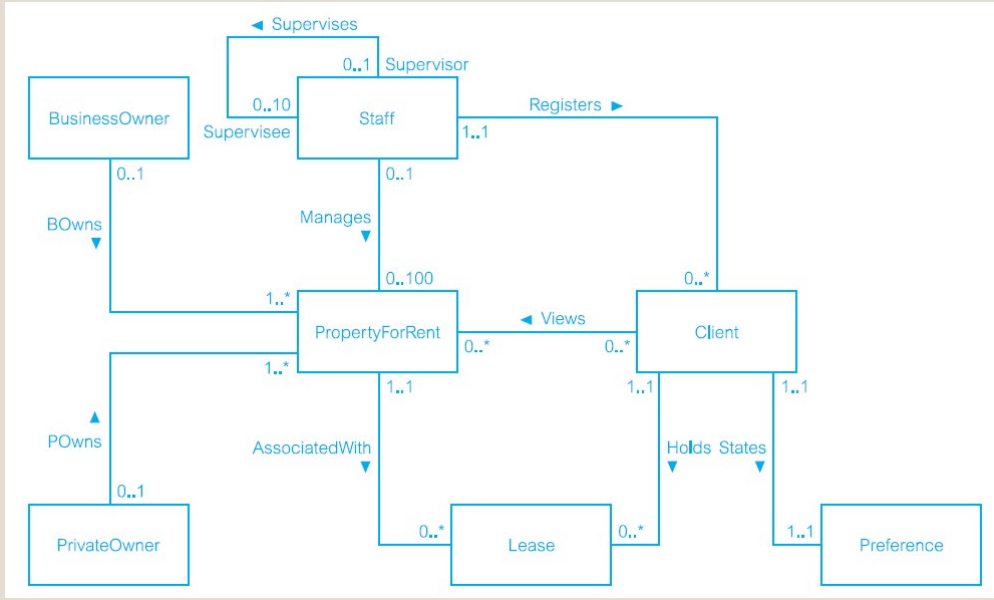
Step 1.1: Identify entity types

- To identify the required entity types by examining the users' requirements specification
- Document entity types

Entity name	*Description*	*Aliases*	*Occurrence*
Staff	General term describing all staff employed by *DreamHome*.	Employee	Each member of staff works at one particular branch.
PropertyForRent	General term describing all property for rent.	Property	Each property has a single owner and is available at one specific branch, where the property is managed by one member of staff. A property is viewed by many clients and rented by a single client, at any one time.

Step 1.2: Identify relationship types

- To identify the important relationships that exist between the entity types.
- Use ERD
- Determine the multiplicity constraints of relationship types
- Check for fan and chasm traps
- Document relationship types



Entity name	*Multiplicity*	*Relationship*	*Multiplicity*	*Entity name*
Staff	0..1 0..1	*Manages* *Supervises*	0..100 0..10	**PropertyForRent** **Staff**
PropertyForRent	1..1	*AssociatedWith*	0..*	**Lease**

Step 1.3 Identify and associate attributes with entity or relationship types

To associate attributes with the appropriate entity or relationship types and document the details of each attribute.

- Simple/composite attributes
- Single/multi-valued attributes
- Derived attributes
- Identify potential problems
- Document attributes

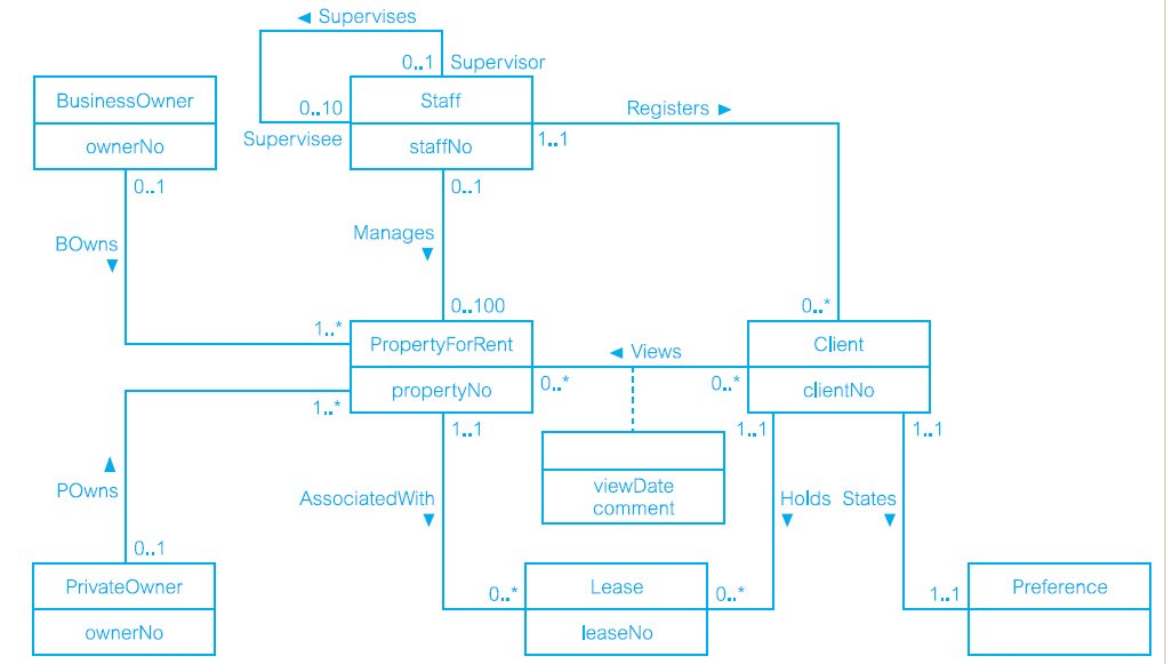
Entity name	*Attributes*	*Description*	*Data Type & Length*	*Nulls*	*Multi-valued*	...
Staff	**staffNo**	Uniquely identifies a member of staff	5 variable characters	No	No	
	name					
	fName	First name of staff	15 variable characters	No	No	
	lName	Last name of staff	15 variable characters	No	No	
	position	Job title of member of staff	10 variable characters	No	No	
	sex	Gender of member of staff	1 character (M or F)	Yes	No	
	DOB	Date of birth of member of staff	Date	Yes	No	
PropertyForRent	**propertyNo**	Uniquely identifies a property for rent	5 variable characters	No	No	

Step 1.4 Determine attribute domains

- To determine domains for the attributes in the data model and document the details of each domain
- Document attribute domains

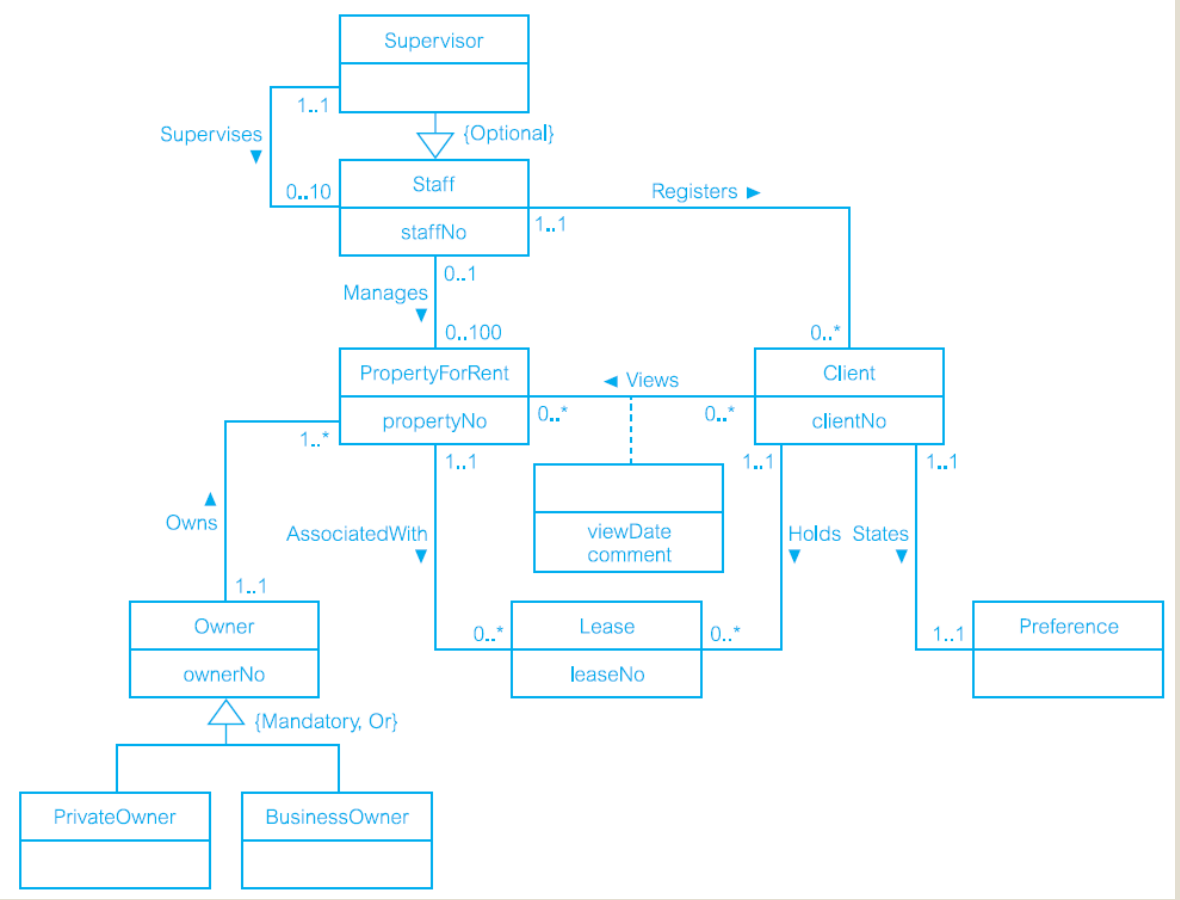
Step 1.5 Determine candidate, primary, and alternate key attributes

- To identify the candidate key(s) for each entity and if there is more than one candidate key, to choose one to be the primary key and the others as alternate keys.
- Document primary and alternate keys



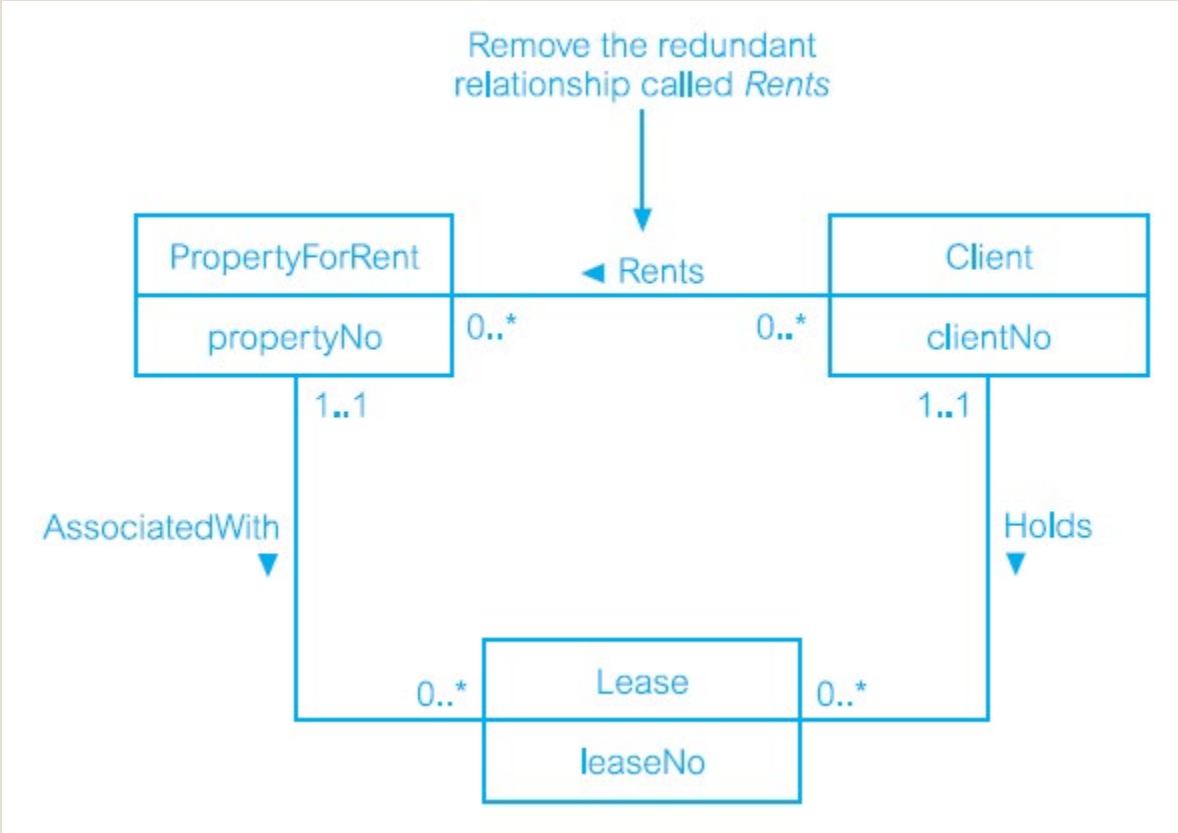
Step 1.6: Consider use of enhanced modeling concepts (optional step)

- To consider the use of enhanced modeling concepts, such as specialization / generalization, aggregation, and composition.



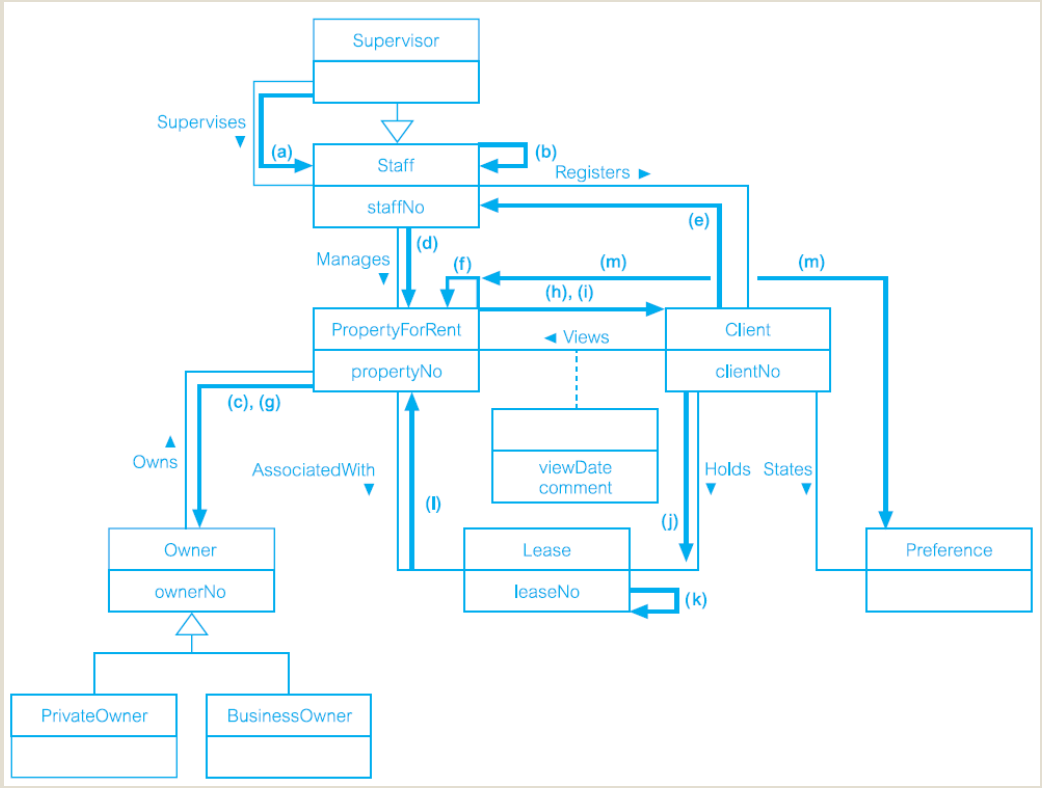
Step 1.7 Check model for redundancy

- To check for the presence of any redundancy in the model and to remove any that does exist.
- Re examine one-to-one relationship
- Remove redundant relationships
- Consider time dimension



Step 1.8 Validate conceptual model against user transactions

- To ensure that the conceptual model supports the required transactions.
- Describing the transaction
- Using transaction pathways



Step 1.9 Review conceptual data model with user

- To review the conceptual data model with the user to ensure that the model is a ‘true’ representation of the data requirements of the enterprise.

STEP 2: Logical Database Design

- The process of constructing a model of the data used in an enterprise based on a specific data model (e.g. relational), but independent of a particular DBMS and other physical considerations.
- Obj: To translate the conceptual data model into a logical data model and then to validate this model to check that it is structurally correct using normalization and supports the required transactions.

Step 2.1 Derive relations for logical data model

Step 2.2 Validate relations using normalization

Step 2.3 Validate relations against user transactions

Step 2.4 Check integrity constraints

Step 2.5 Review logical data model with user

Step 2.6 Merge logical data models into global model (optional step)

Step 2.7 Check for future growth

Step 2.1 Derive relations for logical data model

- Obj: To create relations for the logical data model to represent the entities, relationships, and attributes that have been identified.

Strong entity types	Weak entity types	One-to-many (1:*) binary relationship types
One-to-one (1:1) binary relationship types	One-to-one (1:1) recursive relationship types	Superclass/subclass relationship types
Many-to-many (*:*) binary relationship types	Complex relationship types	Multi-Valued Attributes

Step 2.2 Validate relations using normalization

- To validate the relations in the logical data model using normalization

Step 2.3 Validate relations against user transactions

- To ensure that the relations in the logical data model support the required transactions

Step 2.4 Check integrity constraints

- To check integrity constraints are represented in the logical data model

Required data	Attribute domain constraints	Multiplicity
Entity integrity	Referential integrity	General constraints

Staff (staffNo, fName, lName, position, sex, DOB, supervisorStaffNo)

Primary Key staffNo

Foreign Key supervisorStaffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL

Client (clientNo, fName, lName, telNo, prefType, maxRent, staffNo)

Primary Key clientNo

Foreign Key staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo)

Primary Key propertyNo

Foreign Key ownerNo **references** PrivateOwner(ownerNo) and BusinessOwner(ownerNo)
ON UPDATE CASCADE ON DELETE NO ACTION

Foreign Key staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL

Viewing (clientNo, propertyNo, dateView, comment)

Primary Key clientNo, propertyNo

Foreign Key clientNo **references** Client(clientNo) ON UPDATE CASCADE ON DELETE NO ACTION

Foreign Key propertyNo **references** PropertyForRent(propertyNo)
ON UPDATE CASCADE ON DELETE CASCADE

Lease (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, clientNo, propertyNo)

Primary Key leaseNo

Alternate Key propertyNo, rentStart

Alternate Key clientNo, rentStart

Foreign Key clientNo **references** Client(clientNo) ON UPDATE CASCADE ON DELETE NO ACTION

Foreign Key propertyNo **references** PropertyForRent(propertyNo)
ON UPDATE CASCADE ON DELETE NO ACTION

Step 2.5 Review logical data model with user

- To review the logical data model with the users to ensure that they consider the model to be a true representation of the data requirements of the enterprise.

Step 2.6 Merge logical data models into global Model (optional step)

- To merge logical data models into a single global logical data model that represents all user views of a database.

Step 2.6.1 Merge local logical data models into global model

Step 2.6.2 Validate global logical data model

Step 2.6.3 Review global logical data model with users.

Step 2.7 Check for future growth

- To determine whether there are any significant changes likely in the foreseeable future and to assess whether the logical data model can accommodate these changes.

STEP 3: Physical Database Design

- The process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes design used to achieve efficient access to the data, and any associated integrity constraints and security measures.
- Sources of information for physical design process includes logical data model and documentation that describes model.
- Logical database design is concerned with the what, physical database design is concerned with the how.

Step 3 Translate logical data model for target DBMS

- Step 3.1 Design base relations
- Step 3.2 Design representation of derived data
- Step 3.3 Design general constraints

Step 4 Design file organizations and indexes

- Step 4.1 Analyze transactions
- Step 4.2 Choose file organizations
- Step 4.3 Choose indexes
- Step 4.4 Estimate disk space requirements

Step 5 Design user views

Step 6 Design security mechanisms

Physical Database Design

- To produce a relational database schema from the logical data model that can be implemented in the target DBMS.
- Need to know functionality of target DBMS such as how to create base relations and whether the system supports the definition of:
  - PKs, FKs, and AKs;
  - required data – i.e. whether system supports NOT NULL;
  - domains;
  - relational integrity constraints;
  - general constraints.

Step 3.1 Design base relations

- For each relation, need to define:
  - the name of the relation;
  - a list of simple attributes in brackets;
  - the PK and, where appropriate, AKs and FKs.
  - referential integrity constraints for any FKs identified

```
Domain PropertyNumber:        variable length character string, length 5
Domain Street:                variable length character string, length 25
Domain City:                  variable length character string, length 15
Domain Postcode:              variable length character string, length 8
Domain PropertyType:          single character, must be one of 'B', 'C', 'D', 'E', 'F', 'H', 'M', 'S'
Domain PropertyRooms:         integer, in the range 1–15
Domain PropertyRent:          monetary value, in the range 0.00–9999.99
Domain OwnerNumber:           variable length character string, length 5
Domain StaffNumber:           variable length character string, length 5
Domain BranchNumber:          fixed length character string, length 4
PropertyForRent(
    propertyNo      PropertyNumber      NOT NULL,
    street          Street              NOT NULL,
    city            City                NOT NULL,
    postcode        Postcode,
    type            PropertyType        NOT NULL DEFAULT 'F',
    rooms           PropertyRooms       NOT NULL DEFAULT 4,
    rent            PropertyRent        NOT NULL DEFAULT 600,
    ownerNo         OwnerNumber         NOT NULL,
    staffNo         StaffNumber,
    branchNo        BranchNumber        NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (ownerNo) REFERENCES PrivateOwner(ownerNo) and BusinessOwner(ownerNo)
                                    ON UPDATE CASCADE ON DELETE NO ACTION,
    FOREIGN KEY (branchNo) REFERENCES Branch(branchNo)
                                    ON UPDATE CASCADE ON DELETE NO ACTION);
```

Step 3.1 Design base relations

- For each relation, need to define:
  - the name of the relation;
  - a list of simple attributes in brackets;
  - the PK and, where appropriate, AKs and FKs.
  - referential integrity constraints for any FKs identified
- From data dictionary, we have for each attribute:
- its domain, consisting of a data type, length, and any constraints on the domain;
- an optional default value for the attribute;
- whether it can hold nulls;
- whether it is derived, and if so, how it should be computed.

Step 3.1 Design base relations

```
    Domain PropertyNumber:        variable length character string, length 5
    Domain Street:                variable length character string, length 25
    Domain City:                  variable length character string, length 15
    Domain Postcode:              variable length character string, length 8
    Domain PropertyType:          single character, must be one of 'B', 'C', 'D', 'E', 'F', 'H', 'M', 'S'
    Domain PropertyRooms:         integer, in the range 1–15
    Domain PropertyRent:          monetary value, in the range 0.00–9999.99
    Domain OwnerNumber:           variable length character string, length 5
    Domain StaffNumber:           variable length character string, length 5
    Domain BranchNumber:          fixed length character string, length 4
PropertyForRent(
    propertyNo    PropertyNumber    NOT NULL,
    street        Street            NOT NULL,
    city          City              NOT NULL,
    postcode      Postcode,
    type          PropertyType      NOT NULL DEFAULT 'F',
    rooms         PropertyRooms     NOT NULL DEFAULT 4,
    rent          PropertyRent      NOT NULL DEFAULT 600,
    ownerNo       OwnerNumber       NOT NULL,
    staffNo       StaffNumber,
    branchNo      BranchNumber      NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (ownerNo) REFERENCES PrivateOwner(ownerNo) and BusinessOwner(ownerNo)
                                  ON UPDATE CASCADE ON DELETE NO ACTION,
    FOREIGN KEY (branchNo) REFERENCES Branch(branchNo)
                                  ON UPDATE CASCADE ON DELETE NO ACTION);
```

Step 3.2 Design representation of derived data

- To decide how to represent any derived data present in logical data model in target DBMS.
- Examine logical data model and data dictionary and produce list of all derived attributes.
- Derived attribute can be stored in database or calculated every time it is needed.
- Option selected is based on:
- additional cost to store the derived data and keep it consistent with operational data from which it is derived;
- cost to calculate it each time it is required.
- Less expensive option is chosen subject to performance constraints.

Step 3.2 Design representation of derived data

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Staff

staffNo	fName	lName	branchNo	noOfProperties
SL21	John	White	B005	0
SG37	Ann	Beech	B003	2
SG14	David	Ford	B003	1
SA9	Mary	Howe	B007	1
SG5	Susan	Brand	B003	0
SL41	Julie	Lee	B005	1

Step 3.3 Design general constraints

- To design the general constraints for target DBMS.

- Some DBMS provide more facilities than others for defining enterprise constraints. Example:

```
CONSTRAINT StaffNotHandlingTooMuch
        CHECK (NOT EXISTS (SELECT staffNo
                           FROM PropertyForRent
                           GROUP BY staffNo
                           HAVING COUNT(*) > 100))
```

Step 4: Design File Organizations and Indexes

- To determine the optimal file organizations to store the base relations and the indexes that are required to achieve acceptable performance, that is, the way in which relations and tuples will be held on secondary storage.

Step 4.1 Analyze transactions

- To understand the functionality of the transactions that will run on the database and to analyze the important transactions.

- Attempt to identify performance criteria, such as:
  - transactions that run frequently and will have a significant impact on performance;
  - transactions that are critical to the business;
  - times during the day/week when there will be a high demand made on the database (called the peak load).

Step 4.1 Analyze transactions

- To focus on areas that may be problematic:
  - Map all transaction paths to relations.
  - Determine which relations are most frequently accessed by transactions.
  - Analyze the data usage of selected transactions that involve these relations.

Step 4.1 Analyze transactions

Table 17.1 Cross-referencing transactions and relations.

Transaction/ Relation	(A)				(B)				(C)				(D)				(E)				(F)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Branch										X				X								X		
Telephone																								
Staff		X				X				X								X				X		
Manager																								
PrivateOwner	X																							
BusinessOwner	X																							
PropertyForRent	X					X	X	X						X				X				X		
Viewing																								
Client																								
Registration																								
Lease																								
Newspaper																								
Advert																								

I = Insert; R = Read; U = Update; D = Delete

Step 4.2 Choose file organizations

- To determine an efficient file organization for each base relation.
- File organizations include Heap, Hash, Indexed Sequential Access Method (ISAM), B+-Tree, and Clusters.
- Some DBMSs may not allow selection of file organizations.

Step 4.3 Choose indexes

- To determine whether adding indexes will improve the performance of the system.
- One approach is to keep tuples unordered and create as many secondary indexes as necessary
- Another approach is to order tuples in the relation by specifying a primary or clustering index.

Step 4.4 Estimate disk space requirements

- To estimate the amount of disk space that will be required by the database.

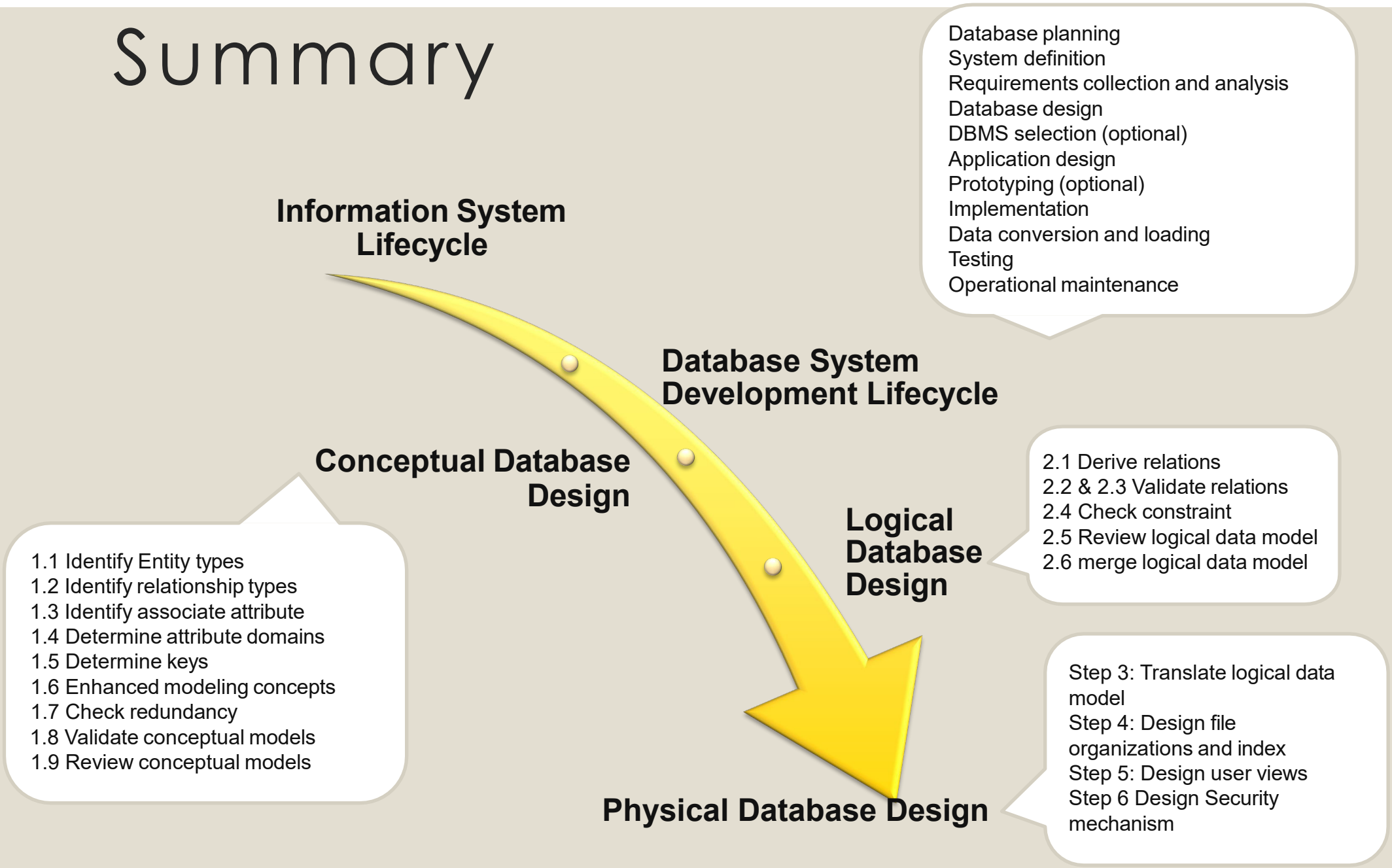
Step 5: Design User Views

- To design the user views that were identified during the Requirements Collection and Analysis stage of the database system development lifecycle.

Step 6: Design Security Measures

- To design the security measures for the database as specified by the users.
- Database security generally provided by DBMS:
  - System security (cover access and use of database at the system level)
  - Data security (cover access and use of database objects and the action that user can have on the objects)

Summary


Information System Lifecycle
Database planning
System definition
Requirements collection and analysis
Database design
DBMS selection (optional)
Application design
Prototyping (optional)
Implementation
Data conversion and loading
Testing
Operational maintenance
Database System Development Lifecycle
Conceptual Database Design
1.1 Identify Entity types
1.2 Identify relationship types
1.3 Identify associate attribute
1.4 Determine attribute domains
1.5 Determine keys
1.6 Enhanced modeling concepts
1.7 Check redundancy
1.8 Validate conceptual models
1.9 Review conceptual models
Logical Database Design
2.1 Derive relations
2.2 & 2.3 Validate relations
2.4 Check constraint
2.5 Review logical data model
2.6 merge logical data model
Physical Database Design
Step 3: Translate logical data model
Step 4: Design file organizations and index
Step 5: Design user views
Step 6 Design Security mechanism

References

- Thomas Connolly and Carolyn Begg, Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition, Pearson, 2015, ISBN: 978- 01329432