UNIVERSITY of HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

COSC 3380 Spring 2024

Database Systems

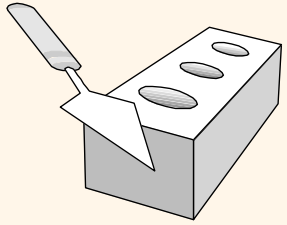
M & W 4:00 to 5:30 PM

Prof. Victoria Hilford

PLEASE TURN your webcam ON (must have)

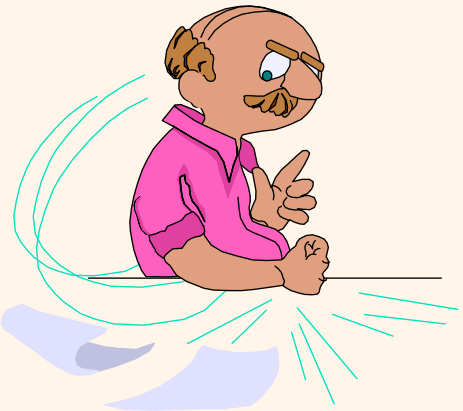
NO CHATTING during LECTURE

VH, UNhide Section 18



COSC 3380

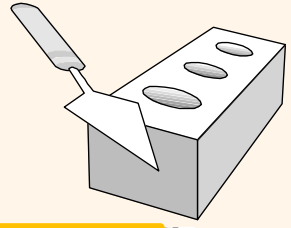
4 to 5:30



PLEASE
LOG IN
CANVAS

Please close all other windows.

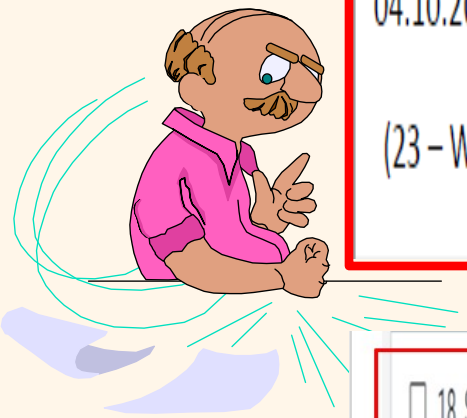
COSC 3380



04.10.2024 (23 – We)	**ZyBook SET 4 - 1**	**Set 4** LECTURE 16 TRANSACTIONS
04.15.2024 (24 – Mo)	**ZyBook SET 4 - 2**	**Set 4** LECTURE 17 CONCURRENCY CONTROL
04.17.2024 (25 – We)	**ZyBook SET 4 - 3**	**Set 4** LECTURE 18 CRASH RECOVERY LECTURE 19 SECURITY and AUTHORIZATION
04.22.2024 (26 – Mo)		**EXAM 4 Practice** **(PART of 20 points)**
04.24.2024 (27 – We)	**TA Download** **ZyBook SET 4 Sections** **(4 PM)** **(PART of 30 points)**	**EXAM 4 Review** **(PART of 20 points)**
04.29.2024 (28 –Mo)		**EXAM 4** **(PART of 50 points)**

COSC 3380

Class 23



04.10.2024	ZyBook SET 4 - 1	Set 4
(23 – We)		LECTURE 16 TRANSACTIONS

18. SET 4 - 1:TRANSACTIONS Hidden 0% 0% v LECTURE 16 TRANSACTIONS I

From 4:00 to 4:07 PM – 5 minutes.

04.10.2024 (23 – We)	ZyBook SET 4 - 1	Set 4 LECTURE 16 TRANSACTIONS

CLASS PARTICIPATION 20 points — 20% of Total

SET 4

Class 23 BEGIN PARTICIPATION
Not available until Apr 10 at 4:00pm | Due Apr 10 at 4:07pm | 100 pts

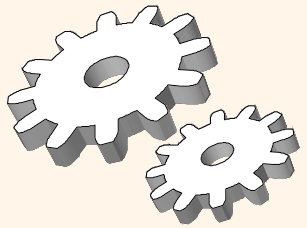
VH, publish

This is a synchronous online class.

Attendance is required.

Recording or distribution of class materials is prohibited.

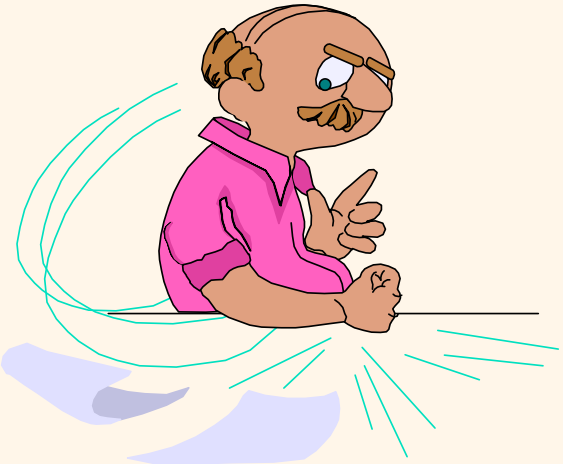
1. At the beginning of selected classes there is an **assessment in the first 10 minutes**. (**beige** BOX in the Detailed Syllabus)
2. At the end of selected classes there is an **assessment in the last 10 minutes**. (**blue** BOX in the Detailed Syllabus)
3. ZyBook sections will be downloaded and used for 30% of Total Score on the dates specified in the Detailed Syllabus.
4. EXAMS are in CANVAS. **No late EXAMS**.
5. I have to be **present in TEAMS in order to take any graded assignment assigned during that class.**



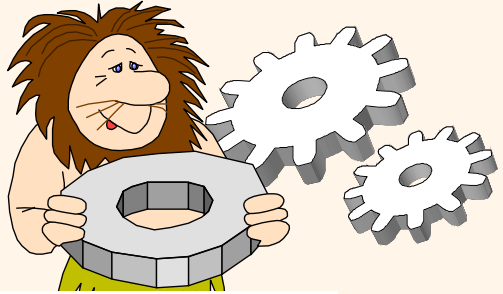
COSC 3380

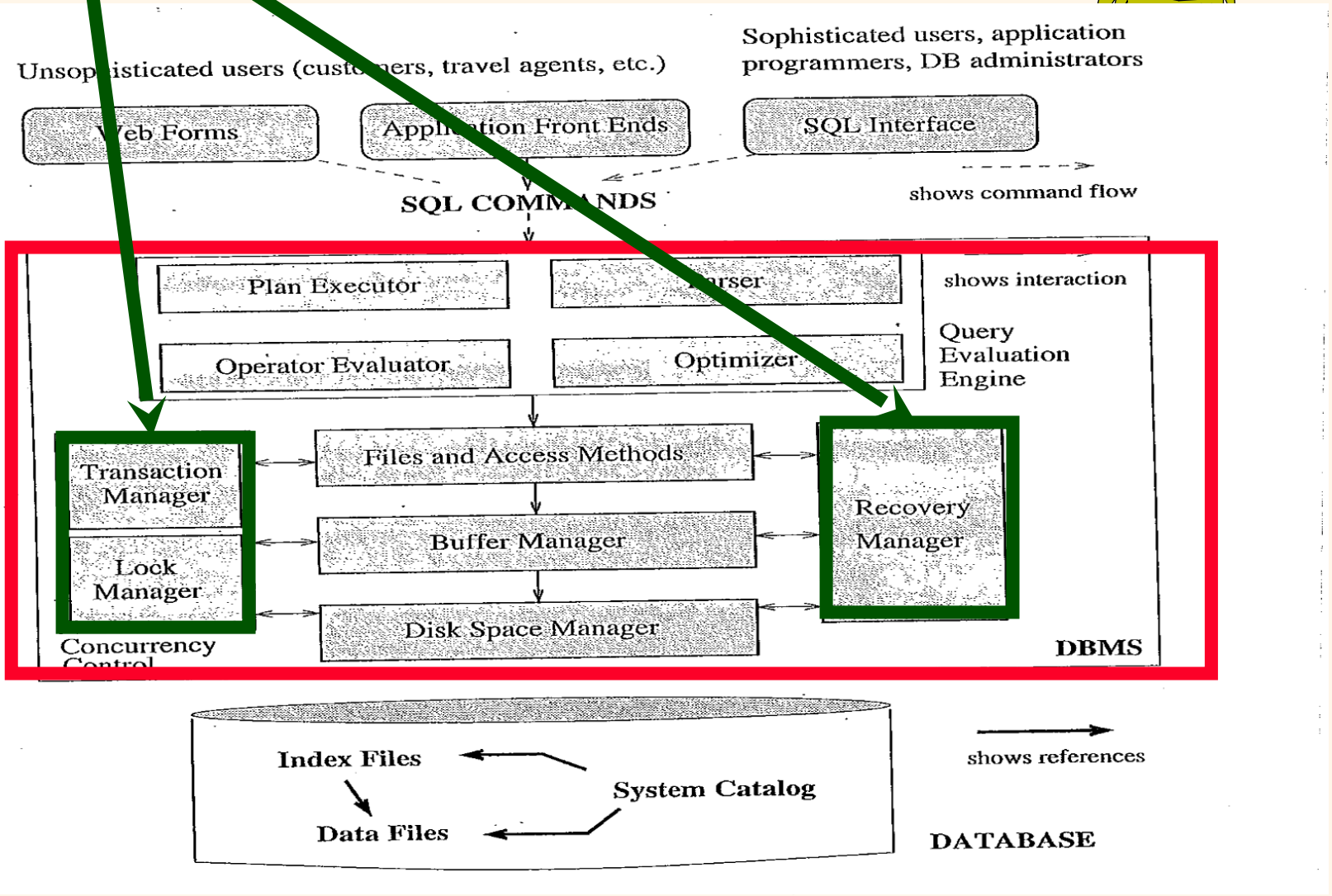
Lecture 16

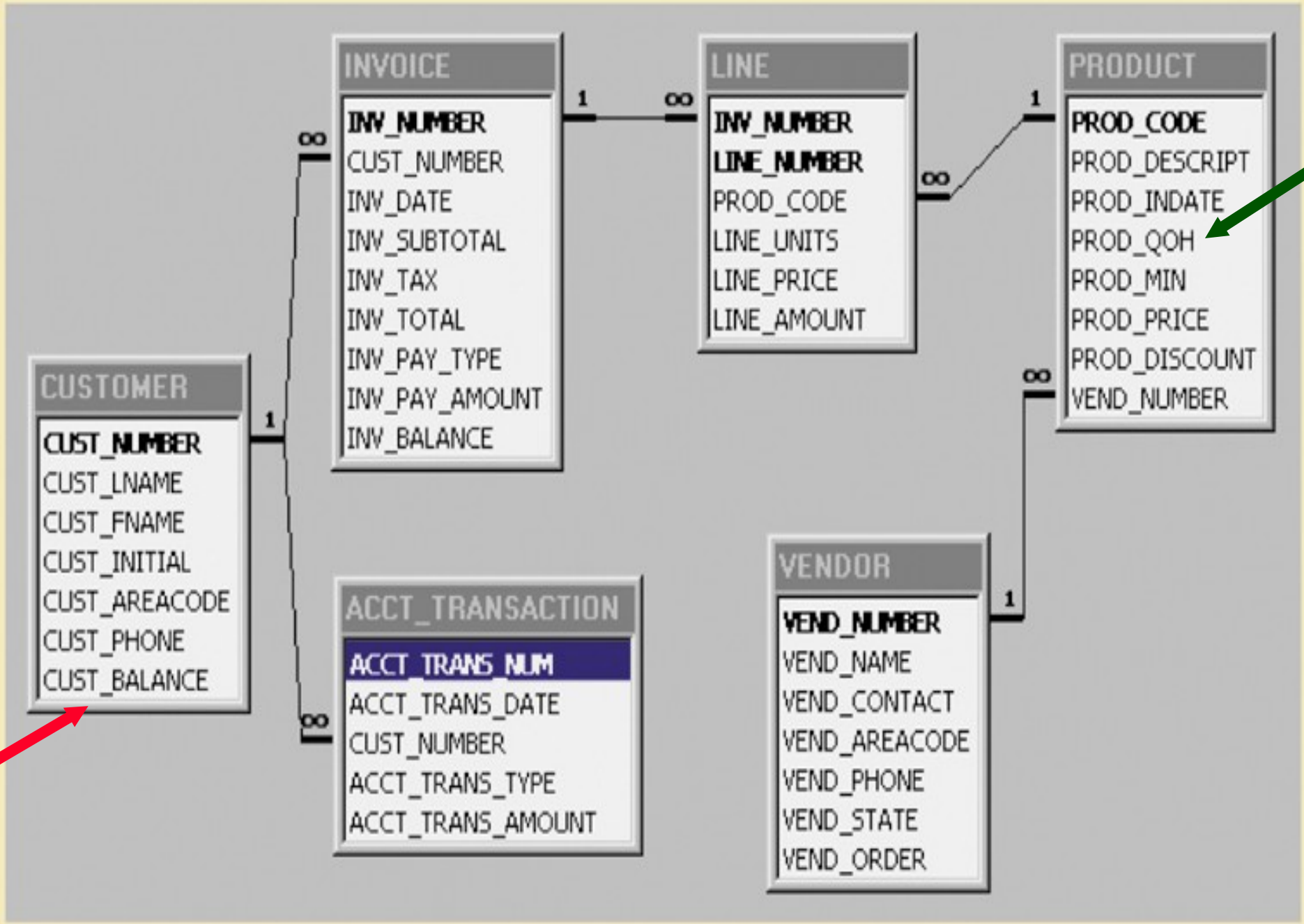
Transaction Management Overview



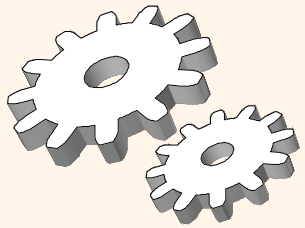
A DBMS




Unsophisticated users (customers, travel agents, etc.)
Sophisticated users, application programmers, DB administrators
Web Forms
Application Front Ends
SQL Interface
SQL COMMANDS
shows command flow
Plan Executor
Parser
shows interaction
Operator Evaluator
Optimizer
Query Evaluation Engine
Transaction Manager
Lock Manager
Files and Access Methods
Buffer Manager
Disk Space Manager
Recovery Manager
Concurrency Control
DBMS
Index Files
System Catalog
Data Files
shows references
DATABASE


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
CUSTOMER
CUST_NUMBER
CUST_LNAME
CUST_FNAME
CUST_INITIAL
CUST_AREACODE
CUST_PHONE
CUST_BALANCE
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
1
∞

Transaction Example - **Notations**



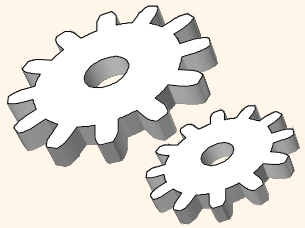
T1 : $r_1(A)$ $w_1(A)$ $r_1(B)$ $w_1(B)$

How many **objects**? 2: object **A** and object **B**

How many actions? 2: **read** and **write**

How many **Transactions**? 1: T_1

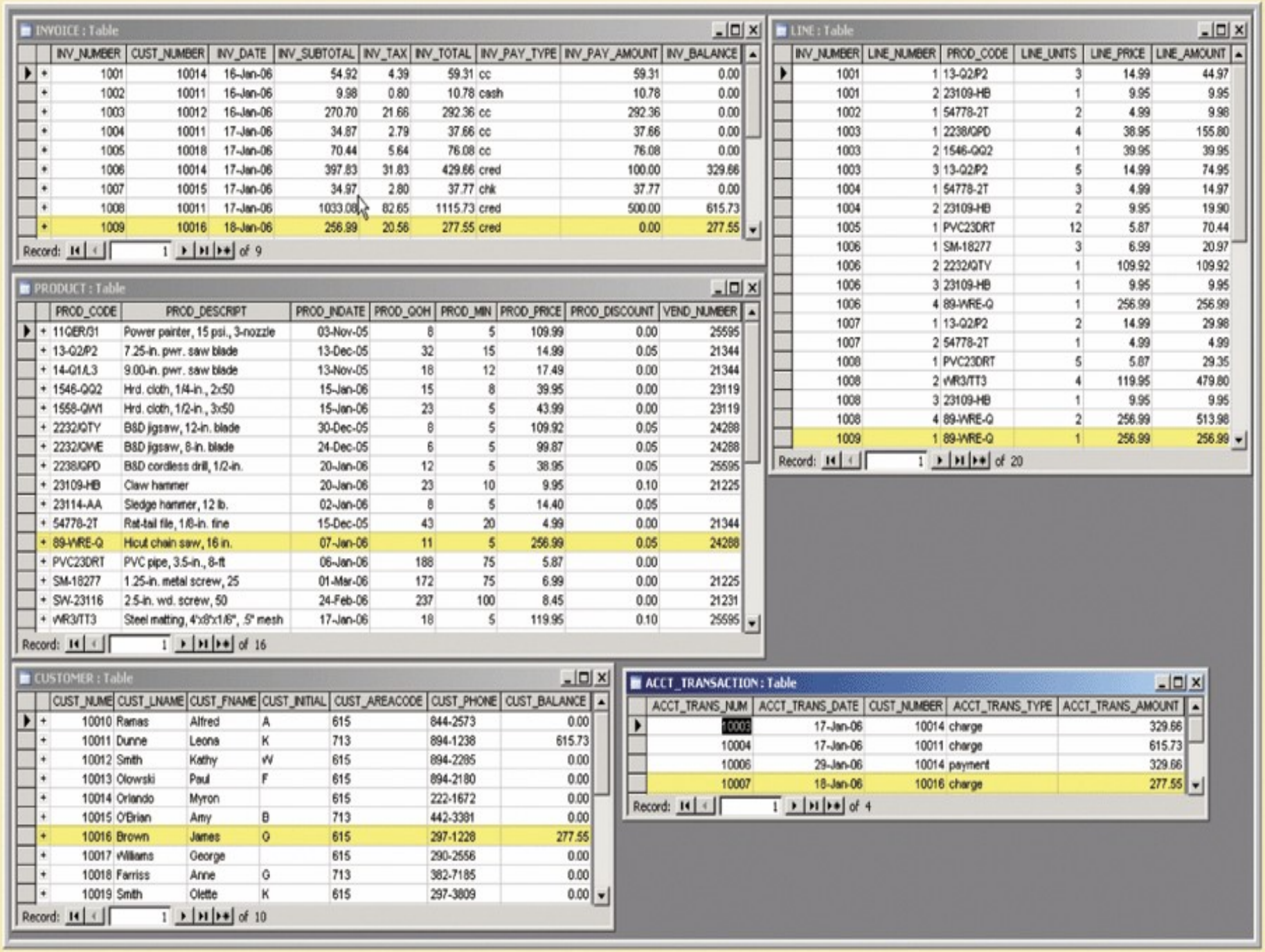
What is a Transaction?



On January 18, 2006 you register the credit sale of one unit of product 89-WRE-Q to customer 10016 in the amount of $277.55.

The SQL statements that represent this **Transaction** are:

```
BEGIN;
    INSERT INTO INVOICE
        VALUES (1009,10016,'18-Jan-2006',256.99,20.56,277.55,'cred',0.00,277.55);
         INSERT INTO LINE
        VALUES (1009, '89-WRE-Q', 1, 256.99,256.99);
         UPDATE PRODUCT
              SET PROD_QOD = PROD_QOH -1
             WHERE PROD_CODE = '89-WRE-Q';
         UPDATE CUSTOMER
        SET CUST_BALANCE = CUST_BALANCE +277.55
            WHERE CUST_NUMBER = 10016;
         INSERT INTO ACCT_TRANSACTION
       VALUES (10007,'18-Jan-06',10016, 'charge',277.55);
COMMIT;
END;
```



INVOICE : Table

INV_NUMBER	CUST_NUMBER	INV_DATE	INV_SUBTOTAL	INV_TAX	INV_TOTAL	INV_PAY_TYPE	INV_PAY_AMOUNT	INV_BALANCE
1001	10014	16-Jan-06	54.92	4.39	59.31	cc	59.31	0.00
1002	10011	16-Jan-06	9.98	0.80	10.78	cash	10.78	0.00
1003	10012	16-Jan-06	270.70	21.66	292.36	cc	292.36	0.00
1004	10011	17-Jan-06	34.87	2.79	37.66	cc	37.66	0.00
1005	10018	17-Jan-06	70.44	5.64	76.08	cc	76.08	0.00
1006	10014	17-Jan-06	397.83	31.83	429.66	cred	100.00	329.66
1007	10015	17-Jan-06	34.97	2.80	37.77	chk	37.77	0.00
1008	10011	17-Jan-06	1033.08	82.65	1115.73	cred	500.00	615.73
1009	10016	18-Jan-06	256.99	20.56	277.55	cred	0.00	277.55

Record: 1 of 9

LINE : Table

INV_NUMBER	LINE_NUMBER	PROD_CODE	LINE_UNITS	LINE_PRICE	LINE_AMOUNT
1001	1	13-Q2/P2	3	14.99	44.97
1001	2	23109-HB	1	9.95	9.95
1002	1	54778-2T	2	4.99	9.98
1003	1	2238/QPD	4	38.95	155.80
1003	2	1546-QQ2	1	39.95	39.95
1003	3	13-Q2/P2	5	14.99	74.95
1004	1	54778-2T	3	4.99	14.97
1004	2	23109-HB	2	9.95	19.90
1005	1	PVC23DRT	12	5.87	70.44
1006	1	SM-18277	3	6.99	20.97
1006	2	2232/QTY	1	109.92	109.92
1006	3	23109-HB	1	9.95	9.95
1006	4	89-WRE-Q	1	256.99	256.99
1007	1	13-Q2/P2	2	14.99	29.98
1007	2	54778-2T	1	4.99	4.99
1008	1	PVC23DRT	5	5.87	29.35
1008	2	VVR3/TT3	4	119.95	479.80
1008	3	23109-HB	1	9.95	9.95
1008	4	89-WRE-Q	2	256.99	513.98
1009	1	89-WRE-Q	1	256.99	256.99

Record: 1 of 20

PRODUCT : Table

PROD_CODE	PROD_DESCRIPT	PROD_INDATE	PROD_QOH	PROD_MIN	PROD_PRICE	PROD_DISCOUNT	VEND_NUMBER
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-05	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-05	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-05	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-06	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-06	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-05	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-05	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-06	12	5	38.95	0.05	25595
23109-HB	Claw hammer	20-Jan-06	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	02-Jan-06	8	5	14.40	0.05	
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-05	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	07-Jan-06	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	06-Jan-06	188	75	5.87	0.00	
SM-18277	1.25-in. metal screw, 25	01-Mar-06	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	24-Feb-06	237	100	8.45	0.00	21231
VVR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-06	18	5	119.95	0.10	25595

Record: 1 of 16

CUSTOMER : Table

CUST_NUMB	CUST_LNAME	CUST_FNAME	CUST_INITIAL	CUST_AREACODE	CUST_PHONE	CUST_BALANCE
10010	Ramas	Alfred	A	615	844-2573	0.00
10011	Dunne	Leona	K	713	894-1238	615.73
10012	Smith	Kathy	W	615	894-2285	0.00
10013	Olowski	Paul	F	615	894-2180	0.00
10014	Orlando	Myron		615	222-1672	0.00
10015	O'Brian	Amy	B	713	442-3381	0.00
10016	Brown	James	G	615	297-1228	277.55
10017	Williams	George		615	290-2556	0.00
10018	Farriss	Anne	G	713	382-7185	0.00
10019	Smith	Olette	K	615	297-3809	0.00

Record: 1 of 10

ACCT_TRANSACTION : Table

ACCT_TRANS_NUM	ACCT_TRANS_DATE	CUST_NUMBER	ACCT_TRANS_TYPE	ACCT_TRANS_AMOUNT
10003	17-Jan-06	10014	charge	329.66
10004	17-Jan-06	10011	charge	615.73
10006	29-Jan-06	10014	payment	329.66
10007	18-Jan-06	10016	charge	277.55

Record: 1 of 4

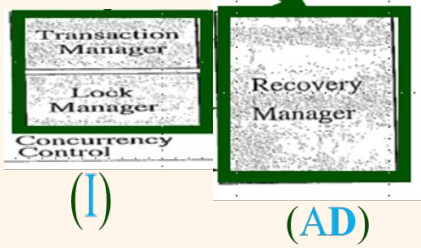
Transaction Properties (**AC**IDS)

- **A**tomicity
  - Requires that **all operations** (**SQL** requests) of a **Transaction** be completed

- **C**onsistency
  - Indicates the **Permanence*** of DATABASE**'s** consistent state

Permanence \Per"ma*nence\, Permanency \Per"ma*nen*cy\, n.
*The quality or state of being permanent; **continuance in the same state or place**; duration; fixedness; as, the permanence of institutions; the permanence of nature.*

Transaction Properties (ACIDS)



- **I**solation
  - **Data** used during execution of a Transaction cannot be used by second Transaction until first one is completed

- **D**urability
  - Indicates **Permanence** of DATABASE's consistent state

Permanence \Per"ma*nence\, Permanency \Per"ma*nen*cy\, n.
The quality or state of being permanent; ***continuance in the same state or place****; duration; fixedness; as, the permanence of institutions; the permanence of nature.*

TA, Fernando (A – L).

TA, Jordan (M – Z).

Please compare CANVAS vs. TEAMS Attendance.
Print screens of students in CANVAS but not in the TEAMS meeting.
(4.10.2024 Attendance X missing LastName.docx)

Anomalies:
Lost Updates or WW Conflict or Overwriting Uncommitted Data

- *Assume that you have a product whose current PROD_QOH value is 35.*
- *Also assume that two **concurrent Transactions**, **T1** and **T2**, occur that update the PROD_QOH value for **same item** in the **PRODUCT** table.*
- The **Transactions** are:

TRANSACTION	COMPUTATION
T1: Purchase 100 units	PROD_QOH = PROD_QOH +100
T2: Sell 30 units	PROD_QOH = PROD_QOH - 30

Anomalies: Lost Updates or WW Conflict or Overwriting Uncommitted Data

Coordination of simultaneous **Transaction** executio in a **multiprocessing** DATABASE **system** (**SERIAL** EXECUTION)

Normal Execution of Two Transactions

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH	135
5	T2	PROD_QOH = 135 – 30	
6	T2	Write PROD_QOH	105

What was written? **105**

Anomalies: Lost Updates or WW Conflict or Overwriting Uncommitted Data

Note that the first Transaction (T1) has not yet been Committed when the second Transaction (T2) is executed. Therefore, **T2 still operates on the value 35**, and its **subtraction yields 5 in memory**. In the meantime, T1 writes the value 135 to disk, which is *promptly overwritten* by T2.

*In short, the addition of 100 units is "**lost**" during the process.*

Lost Updates

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T2	Read PROD_QOH	35
3	T1	PROD_QOH = 35 + 100	
4	T2	PROD_QOH = 35 − 30	
5	T1	Write PROD_QOH (**Lost update**)	135
6	T2	Write PROD_QOH	5

What should be written? **105**

Anomalies: Lost Updates or WW Conflict or Overwriting Uncommitted Data

- Overwriting Uncommitted Data (**WW** Conflicts):

T1:	W(A),		**W**(B), **Commit**
T2:		W(A), **W**(B), **Commit**	

Anomalies:
Uncommitted Data or **WR** Conflict
or **“uncommitted reads”**

*Assume the following Transactions where **T1 updates the inventory***

***T1** is then forced to **roll back** due to an error during the update of the invoice total.*

*This time the **T1** Transaction is **rolled back** to eliminate the addition of the 100 units.*

***Because T2 subtracts 30 from the original 35 units**, the correct answer should be 5!*

TRANSACTION	COMPUTATION
T1: Purchase 100 units	PROD_QOH = PROD_QOH +100 (**ROLLBACK**)
T2: Sell 30 units	PROD_QOH = PROD_QOH - 30

*Anomalies: Uncommitted Data or **WR** conflict or **"uncommitted reads"***

- Under normal circumstances, the **SERIAL** EXECUTION of **T1** and **T2** yields the correct answe (5!).

Correct Execution of Two Transactions

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T1	*****ROLLBACK *****	35
5	T2	Read PROD_QOH	35
6	T2	PROD_QOH = 35 − 30	
7	T2	Write PROD_QOH	5

What was written? **5**

*Anomalies: Uncommitted Data or **WR** conflict or **"uncommitted reads"***

- When uncommitted data (**ROLLBACK is completed after T2 has begun its execution**) is read.

An Uncommitted Data Problem

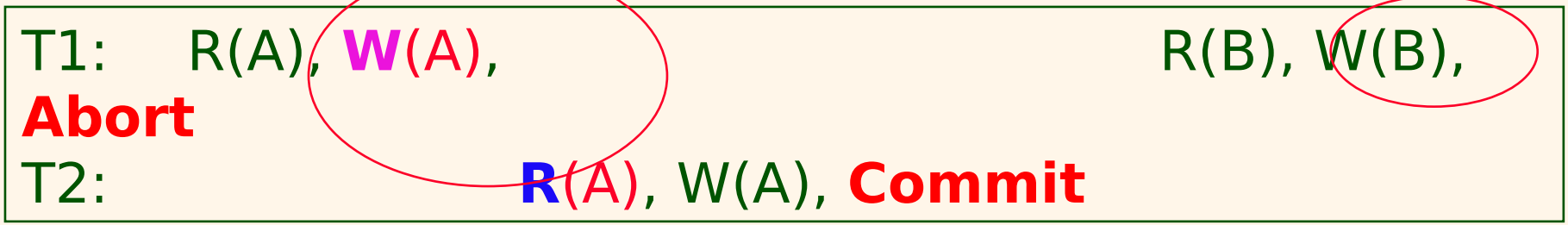
TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH (**Read uncommitted data**)	135
5	T2	PROD_QOH = 135 − 30	
6	T1	***** **ROLLBACK** *****	35
7	T2	Write PROD_QOH	105

What should be written?

5

*Anomalies: Uncommitted Data or **WR** conflict or **“uncommitted reads”***

Reading Uncommitted Data (**WR** Conflicts, “dirty **reads**”):



Isolation

Concurrent transactions T_1 and T_2 can conflict in many ways. Common conflicts include dirty read, nonrepeatable read, and phantom read.

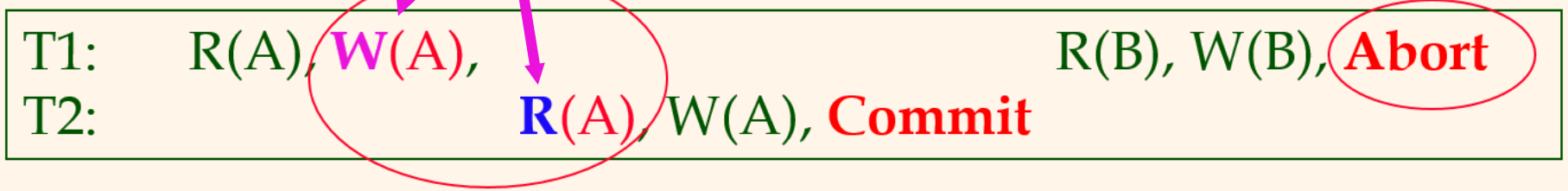
In a ***dirty read***, a transaction reads data that has been updated in a second, uncommitted transaction. The read is invalid If the second transaction either rolls back or makes additional updates to the data. Ex:

1. T_2 updates data X.
2. T_1 reads the updated value of X before T_2 commits.
3. T_2 fails and is rolled back.

Since T_1 reads a value that is eventually rolled back, the result of T_1 is invalid.

Anomalies: Uncommitted Data or WR conflict or "uncommitted reads"

Reading Uncommitted Data (**WR** Conflicts, "dirty **reads**"):



*Anomalies: Inconsistent Retrieval or **RW** Conflict or Unrepeatable **Reads***

- *Assume T1 **calculates** the total quantity on hand of the products stored in the **PRODUCT** table.*

- *At the same time, T2 **updates** **the quantity on hand** (PROD_QOH) for two of the **PRODUCT** table's products.*

*Anomalies: Inconsistent Retrieval or **RW** Conflict or Unrepeatable **Reads***

- *While T1 **calculates** the total quantity on hand for all items, **T2 represents the correction of a typing error**: the **User** added 10 units to product 1558-QW1's PROD_QOH, but meant to add 10 units to product 1546-QQ2's PROD_QOH.*

Retrieval During Update

TRANSACTION T1	TRANSACTION T2
SELECT SUM(PROD_QOH) FROM PRODUCT	UPDATE PRODUCT SET PROD_QOH = PROD_QOH + 10 WHERE PROD_CODE = '1546-QQ2'
	UPDATE PRODUCT SET PROD_QOH = PROD_QOH – 10 WHERE PROD_CODE = '1558-QW1'
	COMMIT;

*Anomalies: Inconsistent Retrieval or **RW** Conflict or Unrepeatable **Reads***

- The correct answer is 92!

Transaction Results: Data Entry Correction

	BEFORE	AFTER
PROD_CODE	PROD_QOH	PROD_QOH
11QER/31	8	8
13-Q2/P2	32	32
1546-QQ2	15	(15 + 10) → 25
1558-QW1	23	(23 − 10) → 13
2232-QTY	8	8
2232-QWE	6	6
Total	92	92

What was written?

92

*Anomalies: Inconsistent Retrieval or **RW** Conflict or Unrepeatable **Reads***

- The computed answer is 102 which is obviously wrong !

Inconsistent Retrievals

TIME	TRANSACTION	ACTION	VALUE	TOTAL
1	T1	Read PROD_QOH for PROD_CODE = '11QER/31'	8	8
2	T1	Read PROD_QOH for PROD_CODE = '13-Q2/P2'	32	40
3	T2	Read PROD_QOH for PROD_CODE = '1546-QQ2'	15	
4	T2	PROD_QOH = 15 + 10		
5	T2	Write PROD_QOH for PROD_CODE = '1546-QQ2'	25	
6	T1	Read PROD_QOH for PROD_CODE = '1546-QQ2'	25	(After) 65
7	T1	Read PROD_QOH for PROD_CODE = '1558-QW1'	23	(Before) 88
8	T2	Read PROD_QOH for PROD_CODE = '1558-QW1'	23	
9	T2	PROD_QOH = 23 − 10		
10	T2	Write PROD_QOH for PROD_CODE = '1558-QW1'	13	
11	T2	***** COMMIT *****		
12	T1	Read PROD_QOH for PROD_CODE = '2232-QTY'	8	96
13	T1	Read PROD_QOH for PROD_CODE = '2232-QWE'	6	102

What should be written? **92**

*Anomalies: Inconsistent Retrieval or **RW** Conflict or Unrepeatable **Reads***

- Unrepeatable **Reads** (**RW** Conflicts):

Anomalies: Inconsistent Retrieval or RW Conflict or Unrepeatable Reads

❖ Unrepeatable Reads (RW Conflicts):

In a ***nonrepeatable read***, a transaction repeatedly reads changing data. Ex:

1. T_1 reads data X.
2. T_2 updates X.
3. T_1 rereads X.

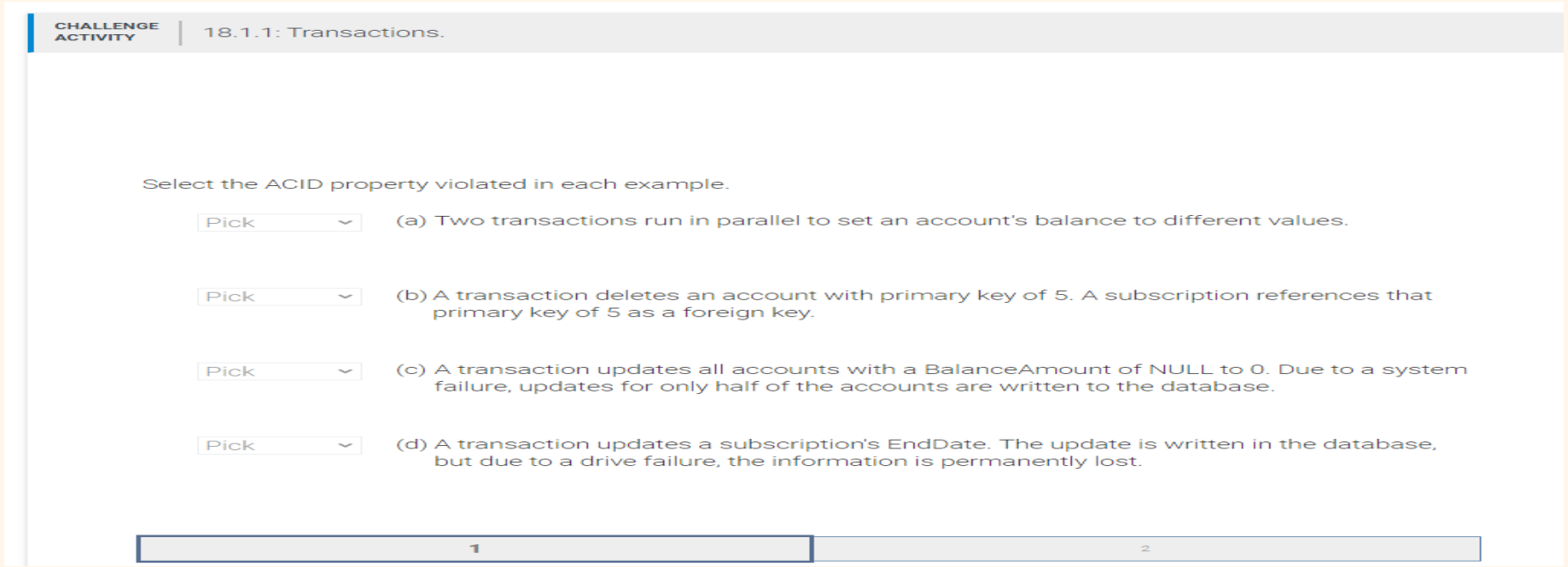
If T_1 incorrectly assumes the value of X is stable, the result of T_1 is invalid.

In a ***phantom read***, one transaction inserts or deletes a table row that another transaction is reading. Ex:

1. T_1 begins reading table rows.
2. T_2 inserts a new row into the table.
3. T_1 continues reading table rows.

Since T_1 sees or misses the new row, depending on precisely when T_2 writes the row to the database, the result of T_1 is unpredictable.

To ensure concurrent transactions are isolated, the concurrency system must prevent dirty reads, nonrepeatable reads, phantom reads, and other potential conflicts. Strict prevention of all conflicts, however, increases transaction duration and resource utilization. Most databases can be configured for relaxed enforcement, producing greater efficiency but occasional violations of isolation.



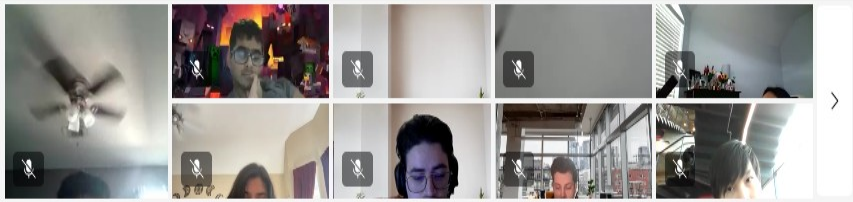
TA time (Fernando)
(CA 18.1.1 Step 1 – Transactions)

TA time (Fernando)
(CA 18.1.1 Step 2 – Transactions)

COSC 3380 23578 M & W 4 to 5:30 PM LECTURE (in TEAMS from 3:50 to 5:15 PM)

40:04

Take control | Pop out | Chat | People 101 | Raise | View | Rooms | Apps | More | Camera | Mic | Share | Leave



FR Ramirez, Ferna... CW Wilkins, Ch... View all

In this meeting (101) Mute all

- Hilford, Victoria — Organizer
- RA Adhikari, Rohit
- MA Ahmed, Mohamed A
- AA Akram, Ali
- BA Akukwe, Benetta O
- SA Alvarez, Stephanie
- OA Anayor-Achu, Ogochukwu E
- HA Avci, Hatice Kubra
- RA Aysola, Riya
- AB Bahl, Anish
- SB Banza, Sean Paolo B
- HB Bui, Hieu
- Burger, Jake
- VC Carrillo-Zepeda, Victor E
- DC Carter, Deric
- CW Christopher Wi... (Unverified)

Arc File Edit View Spaces Tabs Archive Extensions Window Help Wed Apr 10 4:26:06 PM

zyBooks My library > COSC 3380: Database Systems home > 18.1: Transactions zyBooks catalog Help/FAQ Fernando Ramirez

Jump to level 1

Select the ACID property violated in each example.

Durable (a) A transaction updates a subscription's EndDate. The update is written in the database, but due to a drive failure, the information is permanently lost.

Consistent (b) A transaction updates an account with a foreign key. The foreign key does not match any values of the corresponding primary key and is not NULL.

Atomic (c) A transaction removes expired subscriptions. Removes for only half of the subscriptions are written to the database due to a system failure.

Isolated (d) Four transactions run in parallel to reserve the same table for different customers.

1 2

Check Next

✔ Expected: Durable, Consistent, Atomic, Isolated

(a) The drive failure causes changes to be lost after the transaction commits. However, the result of durable transactions must be permanent.

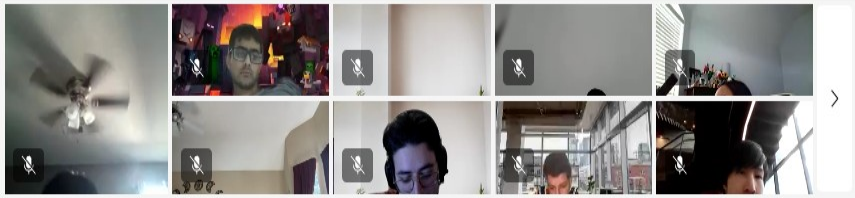
(b) A foreign key that does not match the corresponding primary key violates referential integrity. Referential integrity is a universal rule. The result of a consistent transaction must conform to both universal and business rules.

(c) This transaction writes incomplete results to the database. An atomic transaction must write complete results or none at all.

(d) Isolated transactions run as if no other transactions are running at the same time. Once one transaction reserves the table, the table should be unavailable to any other transaction.

View solution (Instructors only)

Ramirez, Fernando



FR Ramirez, Ferna... CW Wilkins, Ch... View all

Example 1

T_1	T_2
	read all accounts
	compute sum of all accounts
read account C	
increase account C by $5,000	
write account C	
commit	
	read each account
	reduce each account by 0.4% of the sum
	write each account
	commit

Example 2

T_1	T_2
read accounts A through J	
	delete account E
	commit
read accounts K through Z	
compute average of account size	
read all accounts	
add 20% of average to each account	
write each account	
commit	

T_1 no conflict
T_2 nonrepeatable read

T_1 phantom read
T_2 no conflict

1 2

Check Next **Done**. Click any level to practice more. Completion is preserved.

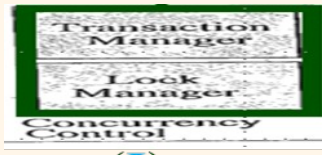
✔ Expected:
Example 1:
T_1 has no conflict
T_2 executes a nonrepeatable read

Example 2:
T_1 executes a phantom read
T_2 has no conflict

In example 1:
T_2 does not write any accounts that are subsequently read by T_1, so T_1 has no conflict.
T_2 reads all accounts once to compute the sum, and again to reduce each account by 0.4% of the sum. In between the two reads, T_1 changes account C. Since T_2 repeatedly reads changing data, T_2 executes a nonrepeatable read.

In example 2:
T_2 deletes account E while T_1 is reading accounts. As a result, the average account size may be invalid. So, T_1 executes a phantom read.
T_1 does not write any accounts that are subsequently read by T_2, so T_2 has no conflict.

Example of Scheduling Transactions : Sa



(I)

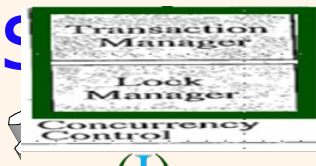
T1	T2	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
Read(B); B ← B+100;			
Write(B);			125
	Read(A);A ←		
A×2;		250	
	Write(A);		
	Read(B);B ←		250
B×2;		250	250
	Write(B);		

time

Sa = $r_1(A)w_1(A)r_1(B)w_1(B)r_2(A)w_2(A)r_2(B)w_2(B)$

Interleaved?

Example of Scheduling Transactions



(I)

Sc

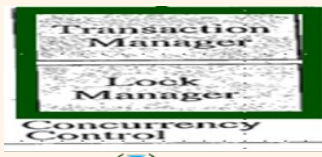
T1	T2
Read(**A**); A ← A+100	
Write(A);	
	Read(A);A ←
A×2;	
	Write(A);
Read(**B**); B ← B+100;	
Write(B);	
	Read(B);B ←
B×2;	
	Write(B);

time

A	B
25	25
125	
250	
	125
	250
250	**250**

$Sc = r_1(\mathbf{A})w_1(A)r_2(A)w_2(A)r_1(\mathbf{B})w_1(B)r_2(B)w_2(B)$

Interleaved?



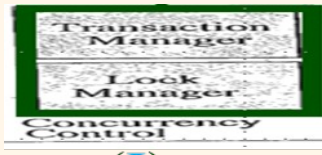
(I)

$Sc = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$

$Sa = r_1(A)w_1(A)\ r_1(B)w_1(B)r_2(A)w_2(A)r_2(B)w_2($ **Serial T_1T_2**

T_1 T_2

- The following **cannot be swapped** without changing the result (conflict)
  - Actions within the **same Transaction**
  - Actions in **different Transactions** ***on the*** *same object* if at least one action is a **write** operation



(I)

Example

- Consider a possible **interleaving** Schedule :

T1:	A=A+100,		B=B -100
T2:		A=1.06*A	B=1.06*B

- This is **OK**. But what about this **Schedule**:

T1:	A=A+100,	B=B -100
T2:	A=1.06*A, B=1.06*B	

- The **DBMS**'s view of the second **Schedule**:

T1:	R(A), W(A),		R(B), W(B)
T2:		R(A), W(A), R(B), W(B)	

This is not **OK**.

Scheduling Transactions (ACIDS)

(I)

- *Serial Schedule:* Schedule that does not interleave the **actions** of different Transactions.

$Sa = \underbrace{r_1(A)w_1(A)\ r_1(B)w_1(B)}_{T_1}\underbrace{r_2(A)w_2(A)r_2(B)w_2(B)}_{T_2}$

- ***Equivalent** Schedules:* For any DATABASE state, **the effect** (on the set of objects in the DATABASE) of executing the first Schedule **is identical** to the effect of executing the second Schedule.

- *Serializable Schedule*: A Schedule that is equivalent Transactions.

$Sc = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$

$Sa = \underbrace{r_1(A)w_1(A)\ r_1(B)w_1(B)}_{T_1}\underbrace{r_2(A)w_2(A)r_2(B)w_2(B)}_{T_2}$

ata

cy,

Equivalent schedules contain the same transactions with all conflicting operations in the same order. **Conflicting schedules** contain the same transactions with some conflicting operations in different order. Equivalent schedules always have the same result. Conflicting schedules can potentially have different results.

Schedule

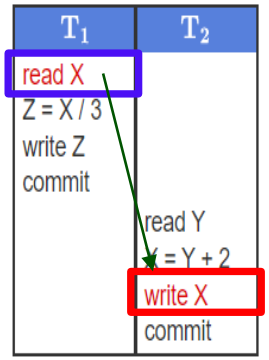
T_1	T_2
read X	
Z = X / 3	
write Z	
commit	
	read Y
	X = Y + 2
	write X
	commit

(R

W)

The schedule has transactions T_1 and T_2, each with a sequence of operations.

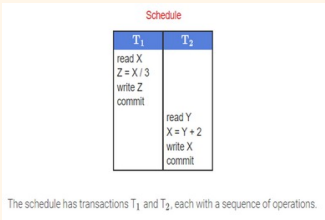
Schedule



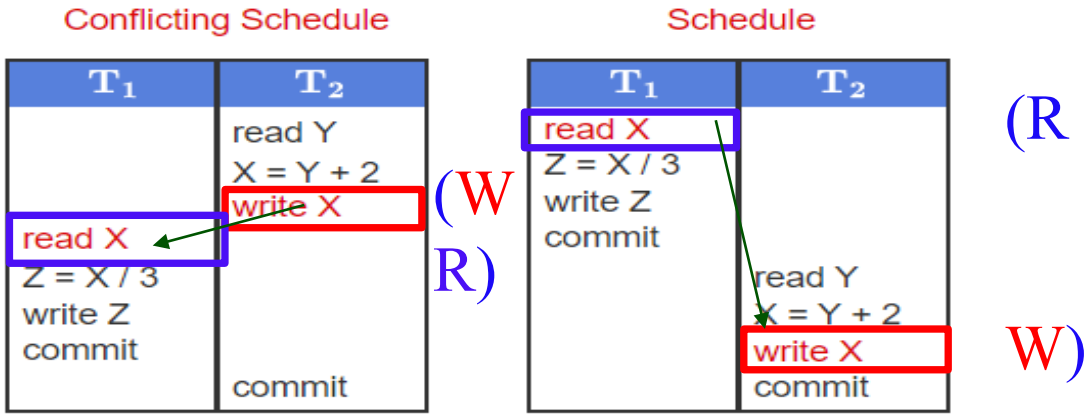
(R

W)

Operations read X and write X conflict because the order of the read and write operations affects the final outcome.

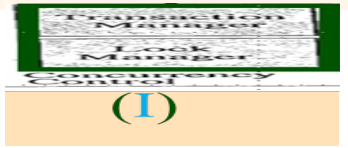


Equivalent schedules contain the same transactions with all conflicting operations in the same order **Conflicting schedules** contain the same transactions with some conflicting operations in different order. Equivalent schedules always have the same result. Conflicting schedules can potentially have different results.



In a conflicting schedule, conflicting operations are in a different order. The final value of Z is different in the conflicting schedule.

Schedules and concurrency



A ***serial schedule*** is a schedule in which transactions are executed one at a time. Serial schedules have no concurrent transactions. Every transaction begins, executes, and commits or rolls back before the next transaction begins. All transactions in a serial schedule are isolated.

Any schedule that is equivalent to a serial schedule is a ***serializable schedule***. A serializable schedule can be transformed into a serial schedule by switching the relative order of reads in different transactions. The order of all operations within a single transaction, and reads and writes in different transactions, cannot be changed.

Serializable schedules generate the same result as the equivalent serial schedule. Therefore, concurrent transactions in a serializable schedule are isolated. (I)

$Sc = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$

$Sa = r_1(A)w_1(A)\ r_1(B)w_1(B)r_2(A)w_2(A)r_2(B)w_2(B)$ Serial T_1T_2

T_1 T_2

- The following **cannot be swapped** without changing the result (conflict)
  - Actions within the **same Transaction**
  - Actions in **different Transactions** ***on the*** *same object* if at least one action is a **write** operation

Isolation levels (I)

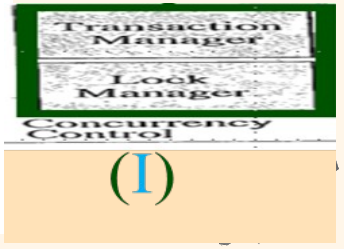
Relational databases allow database administrators and application programmers to specify strict or relaxed levels of isolation for each transaction. The SQL standard defines four isolation levels:

1. **SERIALIZABLE** transactions run in a serializable schedule with concurrent transactions. Isolation is guaranteed.

Table 18.2.1: Isolation levels.

	Phantom read	Nonrepeatable read	Dirty read
SERIALIZABLE	Not allowed	Not allowed	Not allowed

Isolation levels (I)



Relational databases allow database administrators and application programmers to specify strict or relaxed levels of isolation for each transaction. The SQL standard defines four isolation levels:

2. ***REPEATABLE READ*** transactions read only committed data. After the transaction reads data, other transactions **cannot** update the data. REPEATABLE READ prevents most types of isolation violations but allows phantom reads

Table 18.2.1: Isolation levels. (RW) (~~WR~~)

	Phantom read	Nonrepeatable read	Dirty read
REPEATABLE READ	Allowed	Not allowed	Not allowed

(R

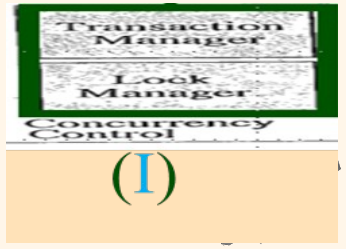
phantom read

T_1	T_2
read first class seats	
	reserve first class seat
read economy seats	
write total seats	
commit	
	commit

W)

Isolation levels (I)

Relational databases allow database administrators and application programmers to specify strict or relaxed levels of isolation for each transaction. The SQL standard defines four isolation levels:



3. ***READ COMMITTED*** transactions read only committed data. After the transaction reads data, other transactions **can** update the data.

READ COMMITTED allows nonrepeatable and phantom reads.

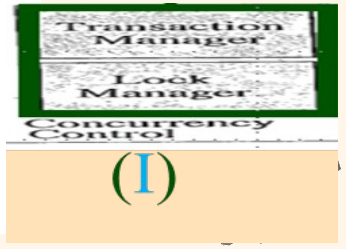
Table 18.2.1: Isolation levels. (RWR)

	Phantom read	Nonrepeatable read	Dirty read
READ COMMITTED	Allowed	Allowed	Not allowed

nonrepeatable read

T_1	T_2
	read seat class
	write ticket cost
upgrade seat	
	read seat class
	assign boarding priority
commit	
	commit

Isolation levels (I)



Relational databases allow database administrators and application programmers to specify strict or relaxed levels of isolation for each transaction. The SQL standard defines four isolation levels:

4. ***READ UNCOMMITTED*** transactions read uncommitted data. READ UNCOMMITTED processes concurrent transactions efficiently but allows a broad range of isolation violations, including dirty, nonrepeatable, and phantom reads.

Table 18.2.1: Isolation levels.

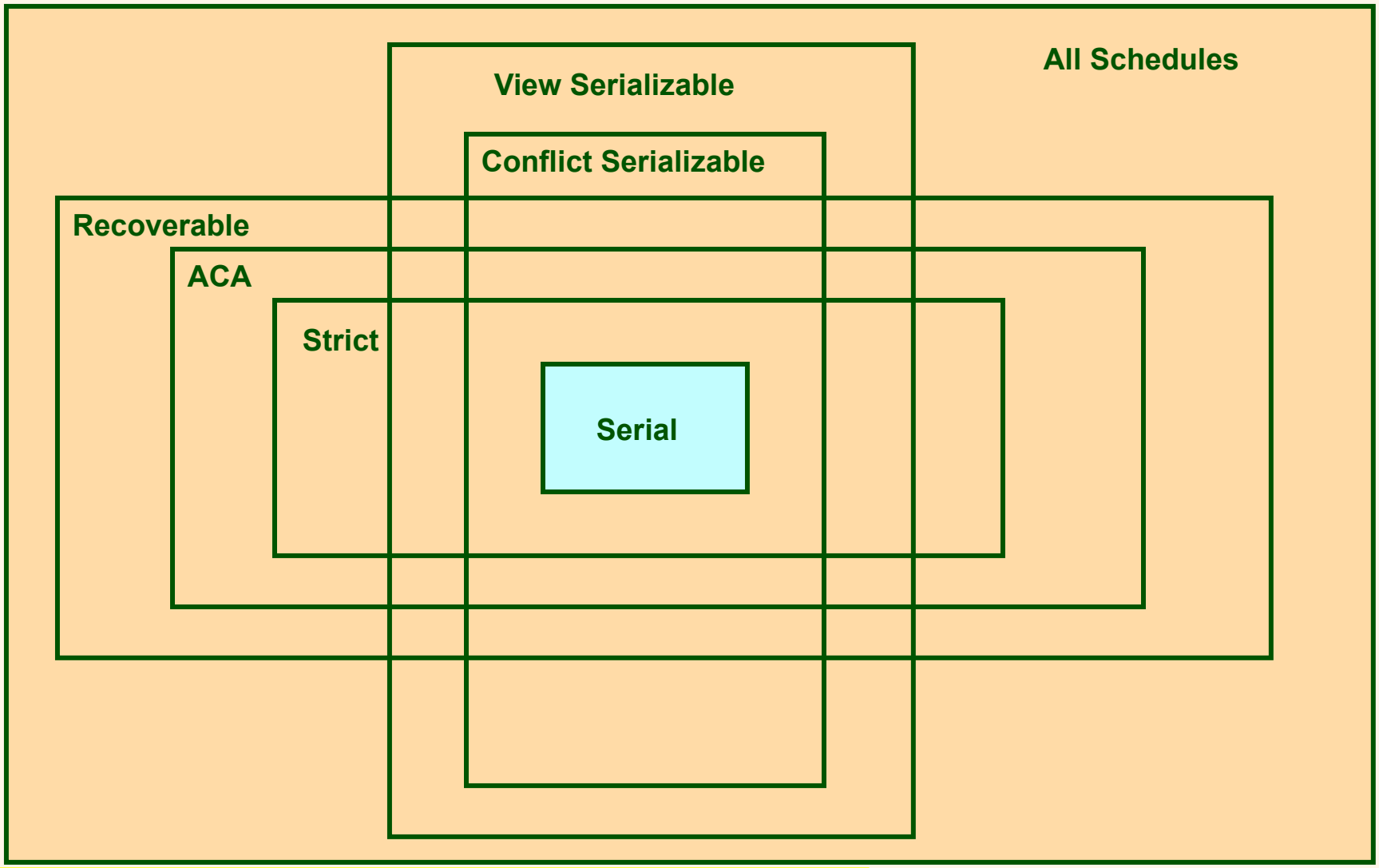
(RWR) (WR)

	Phantom read	Nonrepeatable read	Dirty read
READ UNCOMMITTED	Allowed	Allowed	Allowed

dirty read

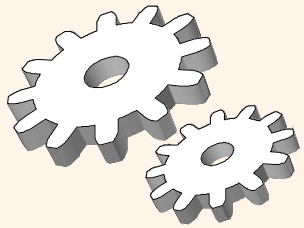
T_1	T_2
upgrade seat	
	read seat class
	write ticket cost
rollback	
	commit

Venn Diagram for Schedules



What is the strongest Schedule?

Venn Diagram for ***Schedules***



Schedules and recovery

(AC)

Serializable schedules affect the concurrency system, which supports isolated transactions. Three additional schedule types affect the recovery system, which supports atomic and durable transactions:

(A D)

- In a ***nonrecoverable schedule***, one or more transactions cannot be rolled back.

Nonrecoverable Schedule

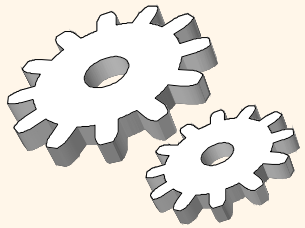
T_1	T_2
write X	
	read X
	commit
~~rollback~~	

(W R)

Recoverable Schedules

Transactions Commit only after all Transactions whose changes they read Commit

Venn Diagram for ***Schedules***



Schedules and recovery

Serializable schedules affect the concurrency system, which supports isolated transactions. Three additional schedule types affect the recovery system, which supports atomic and durable transactions:

- In a **cascading schedule**, rollback of one transaction forces rollback of other transactions.

PARTICIPATION ACTIVITY | 18.2.6: Nonrecoverable, cascading, and strict schedules.

Cascading Schedule

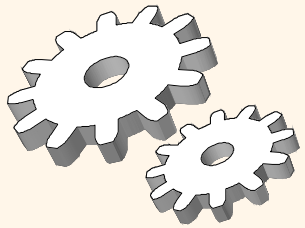
T_1	T_2
write X	
	read X
rollback	
	must rollback

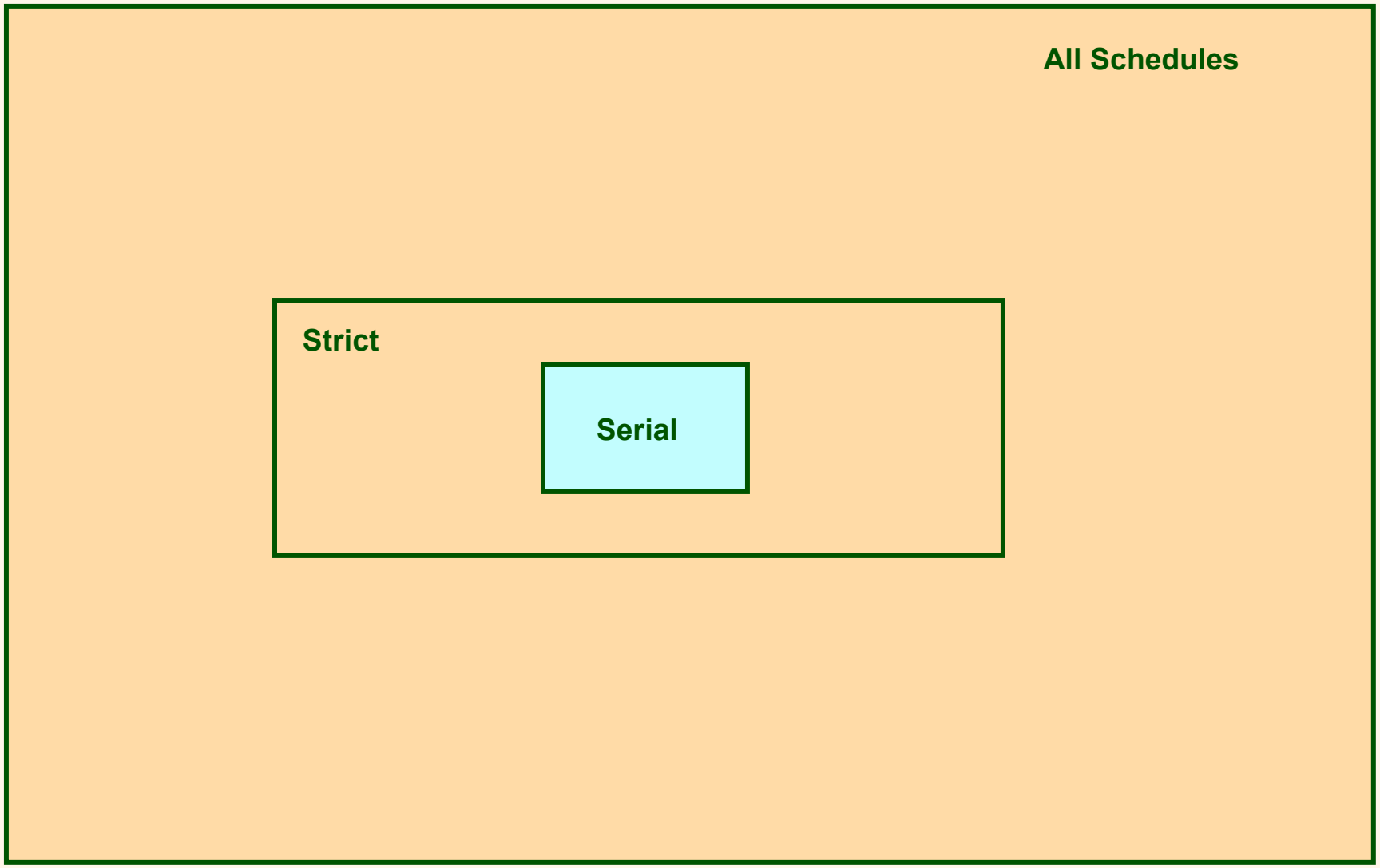
(W R)

Avoids-Cascading-Aborts Schedules

A single Transaction Abort leads to a series of Transactions Rollback. THUS, every Transaction reads only the items that are written by Committed Transactions

Venn Diagram for ***Schedules***





Schedules and recovery

Serializable schedules affect the concurrency system, which supports isolated transactions. Three additional schedule types affect the recovery system, which supports atomic and durable transactions:

In a **strict schedule** rollback of one transaction never forces rollback of other transactions.

PARTICIPATION ACTIVITY | 18.2.6: Nonrecoverable, cascading, and strict schedules.

Strict Schedule

T_1	T_2
write X	
commit	
	read X
	rollback

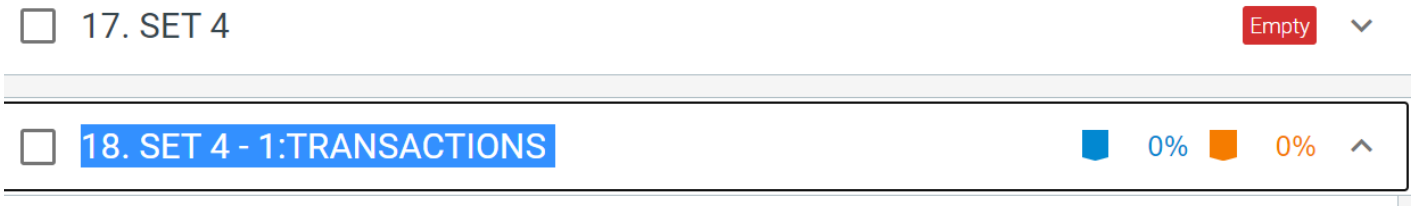
(W
R)

Strict Schedules

If for any two Transactions T1, T2, if a write operation of T1 precedes a conflicting operation of T2, then the Commit event of T1 also precedes that conflicting operation of T2, i.e., a schedule in which a Transaction can neither read or write an item X until the last Transaction that wrote X has Committed.

At 5:00 PM .

04.10.2024 (23 – We)	ZyBook SET 4 - 1	Set 4 LECTURE 16 TRANSACTIONS



VH work on
SET 4 – 1: TRANSACTIONS

Next

04.15.2024 (24 – Mo)	ZyBook SET 4 - 2	Set 4 LECTURE 17 CONCURRENCY CONTROL

VH, unHIDE

From 5:05 to 5:15 PM – 5 minutes.

04.10.2024 (23 – We)	ZyBook SET 4 - 1	Set 4 LECTURE 16 TRANSACTIONS

CLASS PARTICIPATION 20 points — 20% of Total

SET 4

Class 23 END PARTICIPATION
Not available until Apr 10 at 5:05pm | Due Apr 10 at 5:15pm | 100 pts

VH, publish

This is a synchronous online class.

Attendance is required.

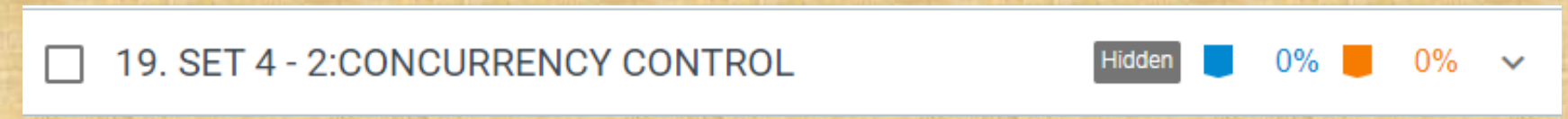
Recording or distribution of class materials is prohibited.

1. At the beginning of selected classes there is an **assessment in the first 10 minutes**. (**beige** BOX in the Detailed Syllabus)
2. At the end of selected classes there is an **assessment in the last 10 minutes**. (**blue** BOX in the Detailed Syllabus)
3. ZyBook sections will be downloaded and used for 30% of Total Score on the dates specified in the Detailed Syllabus.
4. EXAMS are in CANVAS. **No late EXAMS**.
5. I have to be **present in TEAMS in order to take any graded assignment assigned during that class.**

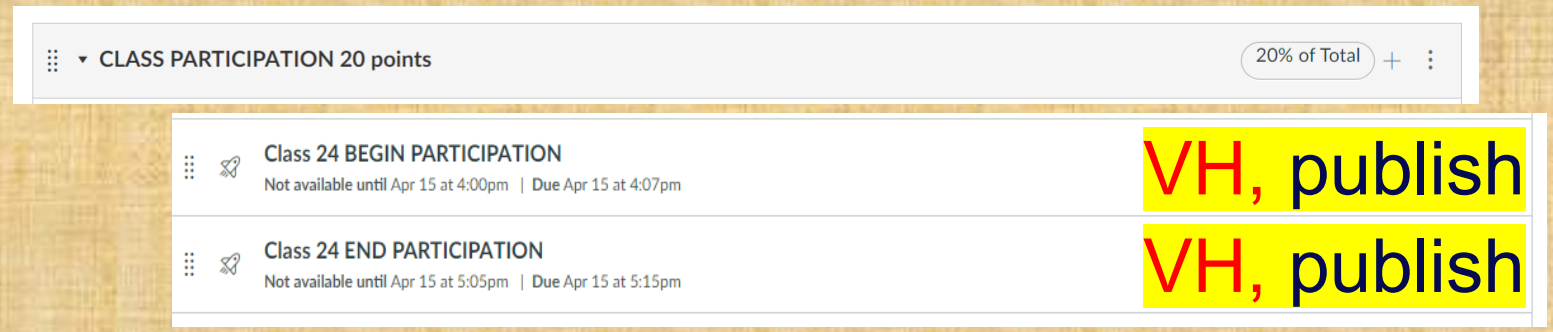
At 5:15 PM.

End Class 23

VH, unHIDE ZyBook SET 4 – 1: TRANSACTIONS.



VH, Download Attendance Report
Rename it:
4.10.2024 Attendance Report FINAL TEAMS



VH, upload Class 23 to CANVAS.