

CS 541
Review

December 6, 2007

Course Outline

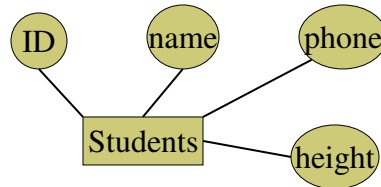
<ol style="list-style-type: none">1. Course Introduction<ul style="list-style-type: none">• Relational Data Model2. Data Modeling<ul style="list-style-type: none">• Entity-Relationship• Constraints3. Relational Theory<ul style="list-style-type: none">• Relational Algebra• Keys and Dependencies• Normalization4. Using a Relational Database<ul style="list-style-type: none">• SQL, Views, Constraints, Triggers5. Storage mechanisms<ul style="list-style-type: none">• I/O cost estimation6. Putting the Data on Disk<ul style="list-style-type: none">• Block/record• Buffer management	<ol style="list-style-type: none">7. Indexing<ul style="list-style-type: none">• Dense vs. Sparse• Index Sequential files• B+-Trees8. Hashing / Bitmap Indexes9. Query Processing10. Query Optimization11. Handling Failure12. Concurrency Control13. Transaction Management14. Research topics (not on final)
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Fall 2007 Chris Clifton - CS541 2

Entity/Relationship Model

Diagrams to represent designs.

- *Entity* like object, = “thing.”
- *Entity set* like class = set of “similar” entities/objects.
- *Attribute* = property of entities in an entity set, similar to fields of a struct.
- In diagrams, entity set → rectangle; attribute → oval.



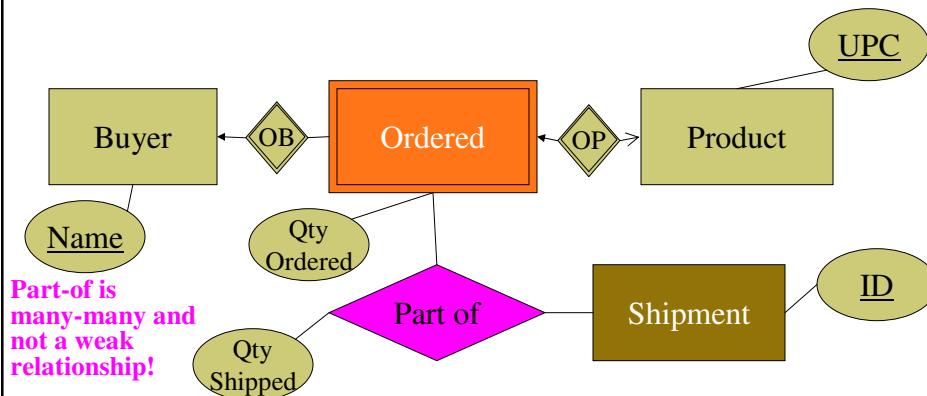
Fall 2007

Chris Clifton - CS541

3

- Complex Example

- ◆ Keys
- ◆ One to many, many to many, one to one relationships
- ◆ Weak entity sets



Fall 2007

Chris Clifton - CS541

4

Relational Model

- Table = *relation*.
- Column headers = *attributes*.
- Row = *tuple*

<u>name</u>	<u>manf</u>
WinterBrew	Pete's
BudLite	A.B.
...	...

Beers

- *Relation schema* = name(attributes) + other structure info., e.g., keys, other constraints. Example: Beers (name, manf)
 - ◆ Order of attributes is arbitrary, but in practice we need to assume the order given in the relation schema.
- *Relation instance* is current set of rows for a relation schema.
- *Database schema* = collection of relation schemas.

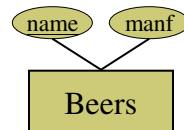
Fall 2007

Chris Clifton - CS541

9

Relational Design

- Simplest approach (not always best): convert each E.S. to a relation and each relationship to a relation.
- Entity Set \rightarrow Relation
- E.S. attributes become relational attributes.



- Becomes:
 - ◆ Beers(name, manf)

Fall 2007

Chris Clifton - CS541

11

Functional Dependencies

$X \rightarrow A$ = assertion about a relation R that whenever two tuples agree on all the attributes of X , then they must also agree on attribute A

Why do we care?

Knowing functional dependencies provides a formal mechanism to divide up relations (*normalization*)

Saves space

Prevents storing data that violates dependencies

Fall 2007

Chris Clifton - CS541

12

Normalization

Goal = BCNF = Boyce-Codd Normal Form = all FD's follow from the fact "key \rightarrow everything."

- Formally, R is in BCNF if for every nontrivial FD for R , say $X \rightarrow A$, then X is a superkey.
 - ◆ "Nontrivial" = right-side attribute not in left side.

Why?

- Guarantees no redundancy due to FD's.
- Guarantees no *update anomalies* = one occurrence of a fact is updated, not all.
- Guarantees no *deletion anomalies* = valid fact is lost when tuple is deleted.

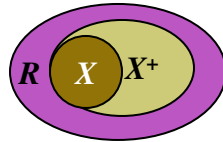
Fall 2007

Chris Clifton - CS541

14

Decomposition to Reach BCNF

- 1. Compute X^+ .
 - ◆ Cannot be all attributes – why?
- 2. Decompose R into X^+ and $(R - X^+) \cup X$.



- 3. Find the FD's for the decomposed relations.
 - ◆ Project the FD's from F = calculate all consequents of F that involve only attributes from X^+ or only from $(R - X^+) \cup X$.

Fall 2007

Chris Clifton - CS541

15

3NF

One FD structure causes problems:

- If you decompose, you can't check all the FD's only in the decomposed relations.
- If you don't decompose, you violate BCNF.

Abstractly: $AB \rightarrow C$ and $C \rightarrow B$.

- Example 1: title city \rightarrow theatre and theatre \rightarrow city.
- Example 2: street city \rightarrow zip, zip \rightarrow city.

Keys: $\{A, B\}$ and $\{A, C\}$, but $C \rightarrow B$ has a left side that is not a superkey.

- Suggests decomposition into BC and AC .
 - ◆ But you can't check the FD $AB \rightarrow C$ in only these relations.

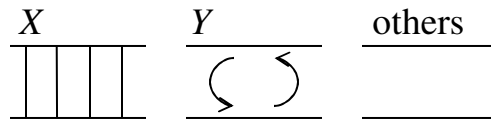
Fall 2007

Chris Clifton - CS541

16

Multivalued Dependencies

The *multivalued dependency* $X \twoheadrightarrow Y$ holds in a relation R if whenever we have two tuples of R that agree in all the attributes of X , then we can swap their Y components and get two new tuples that are also in R .



Fall 2007

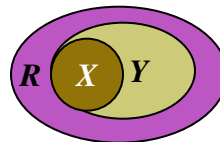
Chris Clifton - CS541

19

4NF

Eliminate redundancy due to multiplicative effect of MVD's.

- Roughly: treat MVD's as FD's for decomposition, but not for finding keys.
- Formally: R is in Fourth Normal Form if whenever MVD $X \twoheadrightarrow Y$ is *nontrivial* (Y is not a subset of X , and $X \cup Y$ is not all attributes), then X is a superkey.
 - ◆ Remember, $X \rightarrow Y$ implies $X \twoheadrightarrow Y$, so 4NF is more stringent than BCNF.
- Decompose R , using 4NF violation $X \twoheadrightarrow Y$, into XY and $X \cup (R - Y)$.



Fall 2007

Chris Clifton - CS541

21

Relational Algebra

- limited expressive power (subset of possible queries)
- good optimizer possible
- rich enough language to express enough useful things

Finiteness

σ SELECT	} UNARY	} FUNDAMENTAL
π PROJECT		
\times CARTESIAN PRODUCT	} BINARY	
\cup UNION		
$-$ SET-DIFFERENCE		
\cap SET-INTERSECTION	} CAN BE DEFINED IN TERMS OF FUNDAMENTAL OPS	
\bowtie_{θ} THETA-JOIN		
\bowtie NATURAL JOIN		
\div DIVISION or QUOTIENT		

Fall 2007 Chris Clifton - CS541 24

Extended (“Nonclassical”) Relational Algebra

Adds features needed for SQL, bags.

1. Duplicate-elimination operator δ .
2. Extended projection.
3. Sorting operator τ .
4. Grouping-and-aggregation operator γ .
5. Outerjoin operator \bowtie_{O} .

Fall 2007 Chris Clifton - CS541 25

SQL

- DML
 - ◆ select, from, where, renaming
 - ◆ set operations
 - ◆ ordering
 - ◆ aggregate functions
 - ◆ nested subqueries
- other parts: DDL, embedded SQL, auth etc

Fall 2007

Chris Clifton - CS541

26

Constraints

Commercial relational systems allow much more “fine-tuning” of constraints than do the modeling languages we learned earlier.

- In essence: SQL programming is used to describe constraints.

Outline

1. Primary key declarations.
2. Foreign-keys = referential integrity constraints.
3. Attribute- and tuple-based checks = constraints within relations.
4. SQL Assertions = global constraints.
 - ◆ Not found in Oracle.
5. Oracle Triggers.
 - ◆ A substitute for assertions.

Fall 2007

Chris Clifton - CS541

27

Triggers (Oracle Version)

Often called event-condition-action rules.

- *Event* = a class of changes in the DB, e.g., “insertions into Beers.”
- *Condition* = a test as in a where-clause for whether or not the trigger applies.
- *Action* = one or more SQL statements.
- Differ from checks or SQL assertions in that:
 1. Triggers invoked by the event; the system doesn’t have to figure out when a trigger could be violated.
 2. Condition not available in checks.

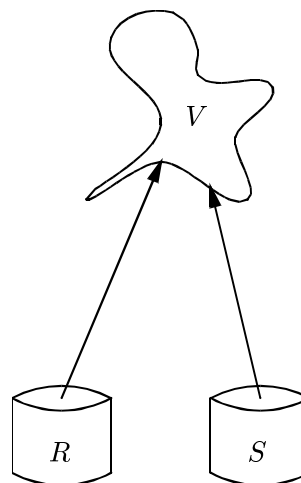
Fall 2007

Chris Clifton - CS541

28

Views

An expression that describes a table without creating it.



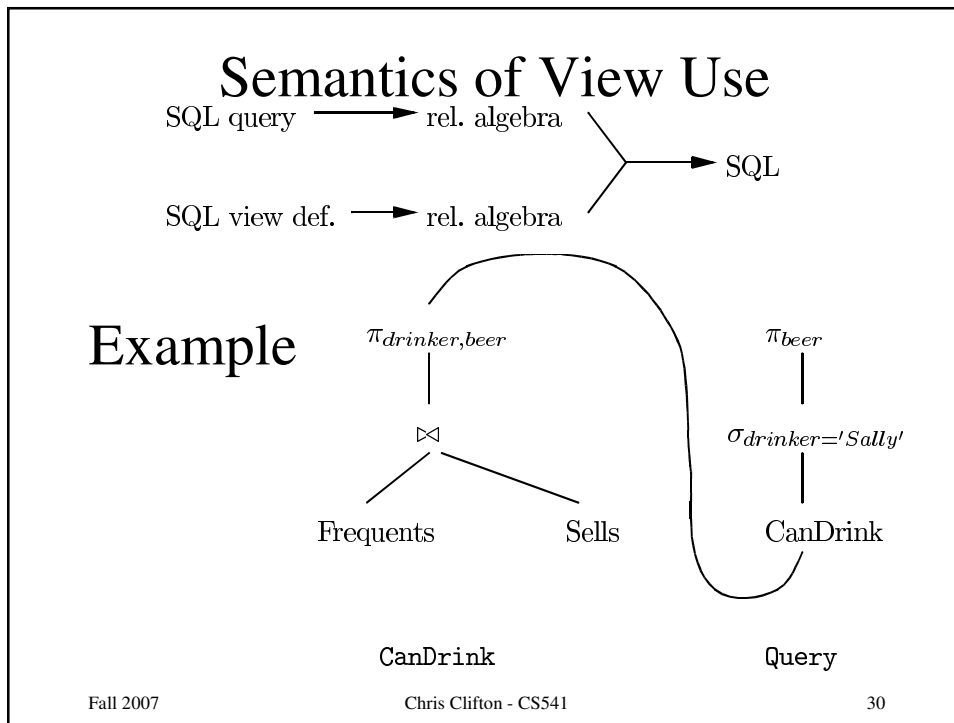
- View definition form is:

```
CREATE VIEW <name> AS <query>;
```

Fall 2007

Chris Clifton - CS541

29

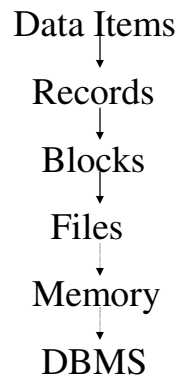


Storage

- Secondary storage, mainly disks
- I/O times
 - Time = Seek Time +
 - Rotational Delay +
 - Transfer Time +
 - Other
- I/Os should be avoided,
especially random ones.....

Fall 2007 Chris Clifton - CS541 34

- How to lay out data on disk



Fall 2007

Chris Clifton - CS541

35

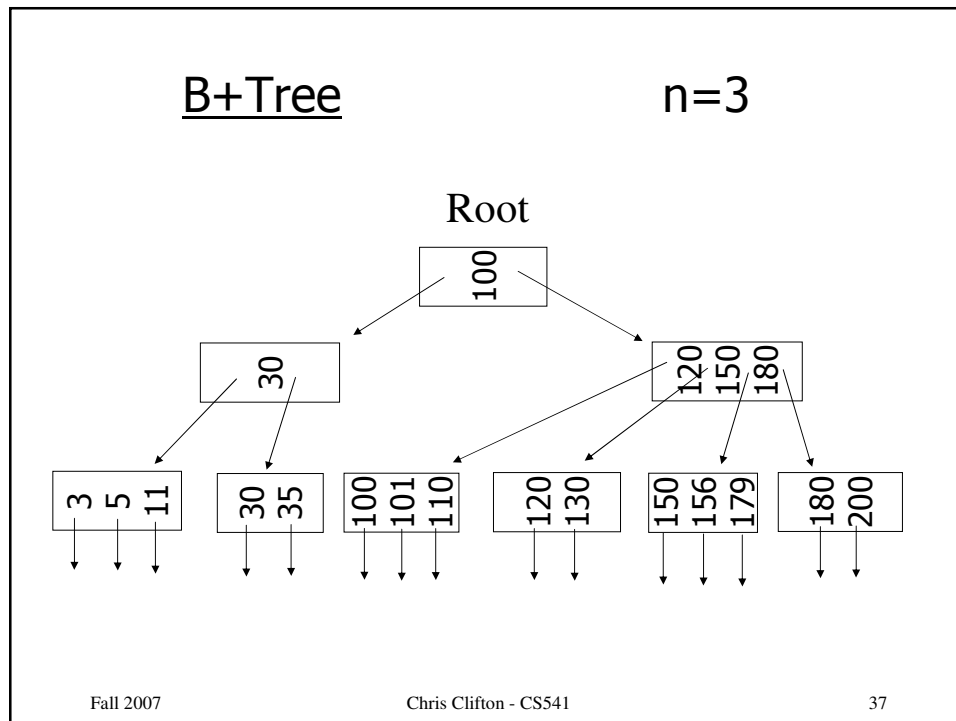
Indexing

- Index sequential file
- Search key (\neq primary key)
- Primary index (on Sequencing field)
- Secondary index
- Dense index (all Search Key values in)
- Sparse index
- Multi-level index
- Insertion/deletion

Fall 2007

Chris Clifton - CS541

36



- B+ tree rules
- (1) All leaves at same lowest level
(balanced tree)
 - (2) Pointers in leaves point to records
except for “sequence pointer”
- Use at least
- Non-leaf: $\lceil (n+1)/2 \rceil$ pointers
- Leaf: $\lfloor (n+1)/2 \rfloor$ pointers to data
- Fall 2007
Chris Clifton - CS541
38



CS 541 Review

Undo/Redo Logging?
Normalization?
Multiple Granularity Locking?
December 6, 2002

undo/redo logging

Update \Rightarrow $\langle T_i, X_{id}, \text{New } X \text{ val}, \text{Old } X \text{ val} \rangle$
page X

- Page X can be flushed before or after T_i commit
- Log record flushed before corresponding updated page (WAL)
- Flush at commit (log only)

Non-quiesce checkpoint

L
O
G

...	Start-ckpt active TR: T ₁ , T ₂ ,	end ckpt	...
-----	-------------------------------------------------------------------	-----	-------------	-----

⋮
for
undo

←→

dirty buffer
pool pages
flushed

Fall 2007

Chris Clifton - CS541

41

Examples what to do at recovery time?

L
O
G

...	T ₁ - a	...	Ckpt T ₁	...	Ckpt end	...	T ₁ - b	
-----	-----------------------	-----	------------------------	-----	-------------	-----	-----------------------	--

no T₁ commit

➤ Undo T₁ (undo a,b)

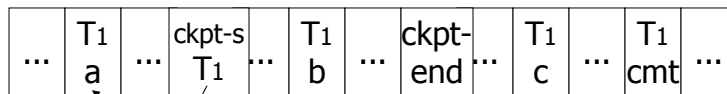
Fall 2007

Chris Clifton - CS541

42

Example

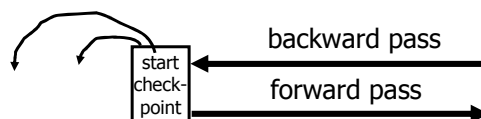
L
O
G



➤ Redo T1: (redo b,c)

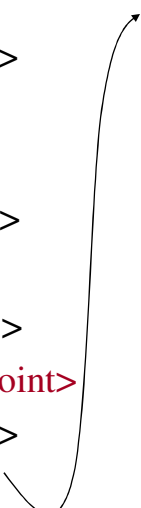
Recovery process:

- Backwards pass (end of log → latest checkpoint start)
 - ◆ construct set S of committed transactions
 - ◆ undo actions of transactions not in S
- Undo pending transactions
 - ◆ follow undo chains for transactions in (checkpoint active list) - S
- Forward pass (latest checkpoint start → end of log)
 - ◆ redo actions of S transactions



Undo/Redo Logging, 17.4.5 (c)

Where could End CKPT be written?

- | | |
|-----------------------|--------------------|
| 1. <Start S> | 10. <Start V> |
| 2. <S, A, 60, 61> | 11. <U, D, 40, 41> |
| 3. <Commit S> | 12. <V, F, 70, 71> |
| 4. <Start T> | 13. <Commit U> |
| 5. <T, A, 61, 62> | 14. <T, E, 50, 51> |
| 6. <Start U> | 15. <Commit T> |
| 7. <U, B, 20, 21> | 16. <V, B, 21, 22> |
| 8. <start checkpoint> | 17. <Commit V> |
| 9. <T, C, 30, 31> | |
- 

Fall 2007

Chris Clifton - CS541

45

Normalization (3.6.1 (e))

- R(A,B,C,D,E)
 - ◆ FDs AB C, C D, D B, D E
- BCNF Violations
- BCNF
- 3NF

Fall 2007

Chris Clifton - CS541

46

Multiple Granularity Locking

Comp		Requestor					
		IS	IX	S	SIX	X	
	Holder	IS	T	T	T	T	F
		IX	T	T	F	F	F
		S	T	F	T	F	F
		SIX	T	F	F	F	F
		X	F	F	F	F	F

Fall 2007
Chris Clifton - CS541
47

Parent locked in	Child can be locked in
IS	IS, S
IX	IS, S, IX, X, SIX
S	[S, IS] not necessary
SIX	X, IX, [SIX]
X	none

Fall 2007
Chris Clifton - CS541
48

Rules

- (1) Follow multiple granularity comp function
- (2) Lock root of tree first, any mode
- (3) Node Q can be locked by T_i in S or IS only if parent(Q) locked by T_i in IX or IS
- (4) Node Q can be locked by T_i in X,SIX,IX only if parent(Q) locked by T_i in IX,SIX
- (5) T_i is two-phase
- (6) T_i can unlock node Q only if none of Q's children are locked by T_i

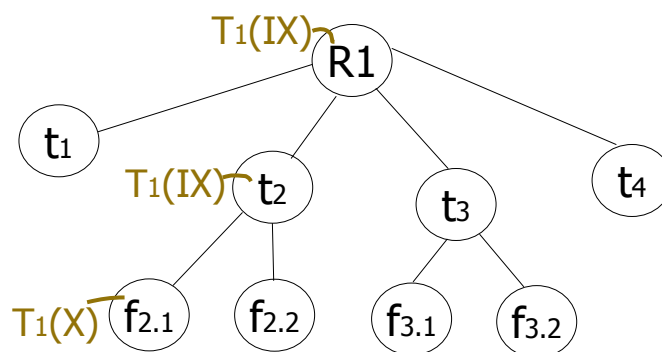
Fall 2007

Chris Clifton - CS541

49

Exercise:

- Can T_2 access object $f_{2.2}$ in X mode? What locks will T_2 get?



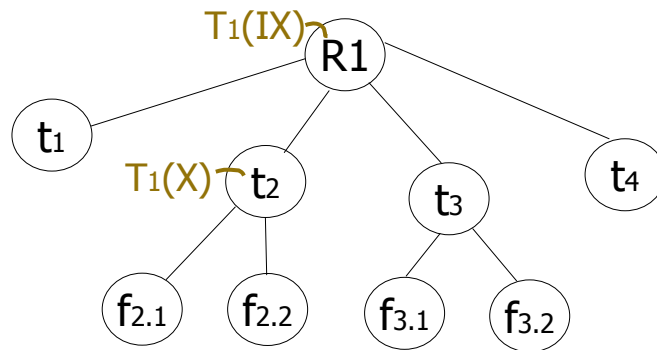
Fall 2007

Chris Clifton - CS541

50

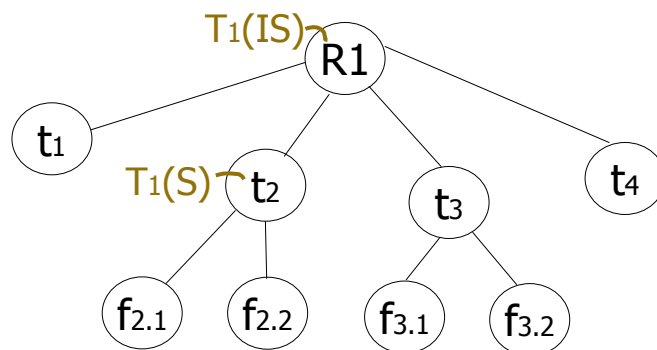
Exercise:

- Can T2 access object f2.2 in X mode? What locks will T2 get?



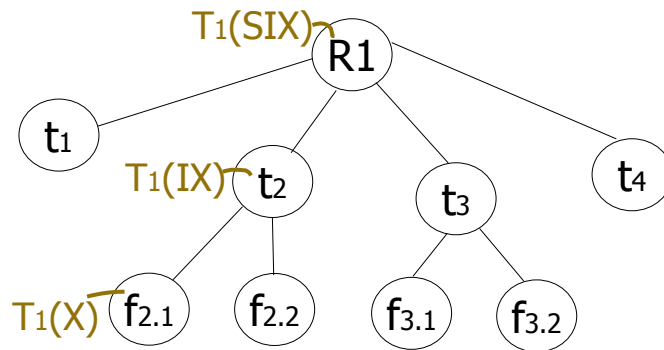
Exercise:

- Can T2 access object f3.1 in X mode? What locks will T2 get?



Exercise:

- Can T2 access object f2.2 in S mode? What locks will T2 get?



Exercise:

- Can T2 access object f2.2 in X mode? What locks will T2 get?

