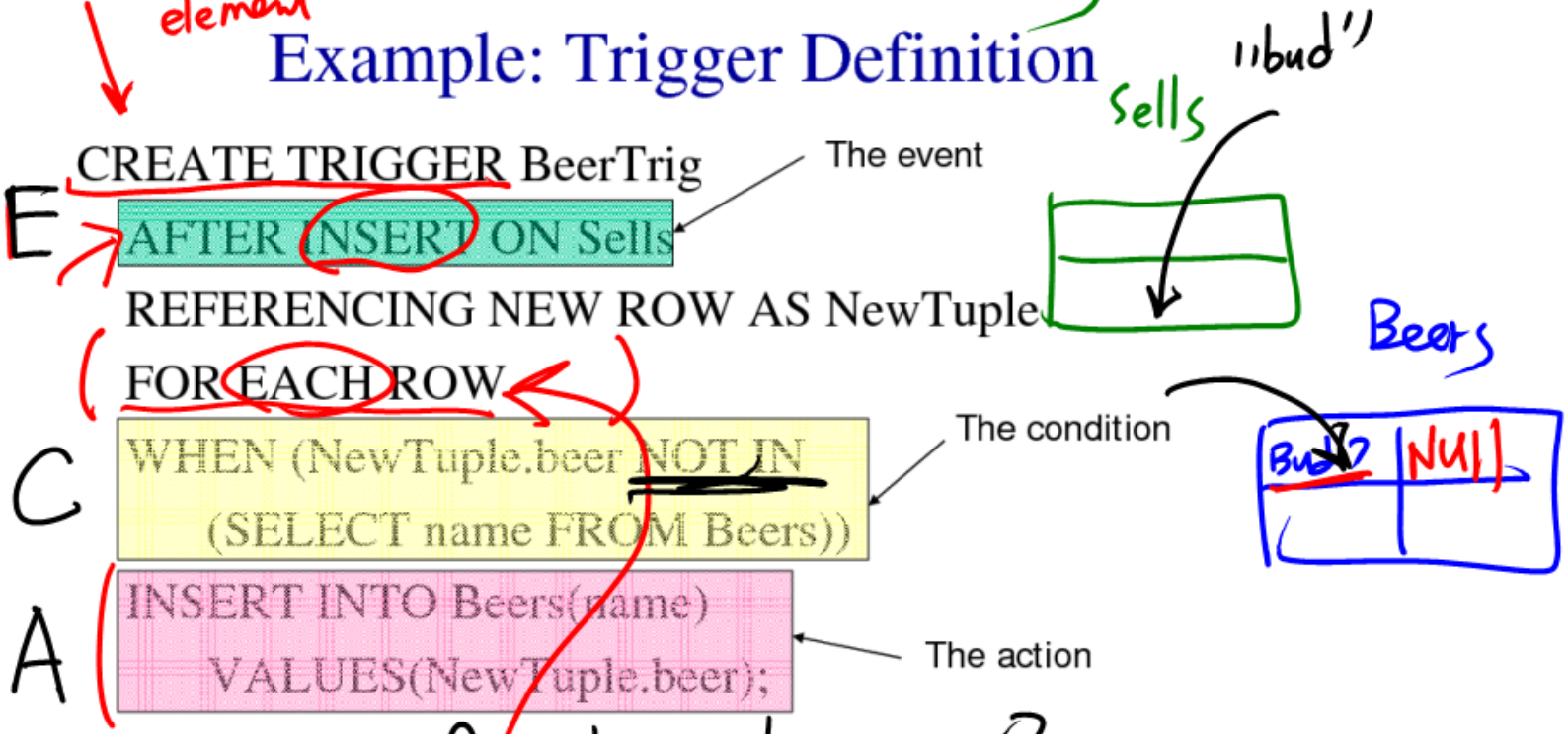
schema element ⇒ create table R (...)

Example: Trigger Definition



Tuple-level What happen?

if no, ⇒ reject.

Options: The Event

- AFTER can be BEFORE.
  - Also, INSTEAD OF if the relation is a view.
    - A great way to execute view modifications: have triggers translate them to appropriate modifications on the base tables.
- INSERT can be DELETE or UPDATE.
  - And UPDATE can be UPDATE . . . ON a particular attribute.

insert into Sells values (. . , . . , . .)

Options: FOR EACH ROW

Tuple-level

- Triggers are either *row-level* or *statement-level*.
- FOR EACH ROW indicates row-level; its absence indicates statement-level.
- Row level triggers are executed once for each modified tuple.
- Statement-level triggers execute once for an SQL statement, regardless of how many tuples are modified.

if no

↓ the SQL stmt.

stmt-level = INSERT/DELETE/update = OLD/NEW table.

Options: REFERENCING

Tuple-Level (for each row)

- INSERT statements imply a new tuple ~~(for row-level) or new set of tuples (for statement-level)~~.
- DELETE implies an old tuple ~~or table~~.
- UPDATE implies both. old/new
- Refer to these by

[NEW OLD][~~TUPLE~~ ROW TABLE] AS <name>

referencing OLD table as OLDT.

Options: The Condition

- Any boolean-valued condition is appropriate.
- It is evaluated before or after the triggering event, depending on whether BEFORE or AFTER is used in the event.
- Access the new/old tuple or set of tuples through the names declared in the REFERENCING clause. (or fixed by "OLD", "NEW" in MySQL.)

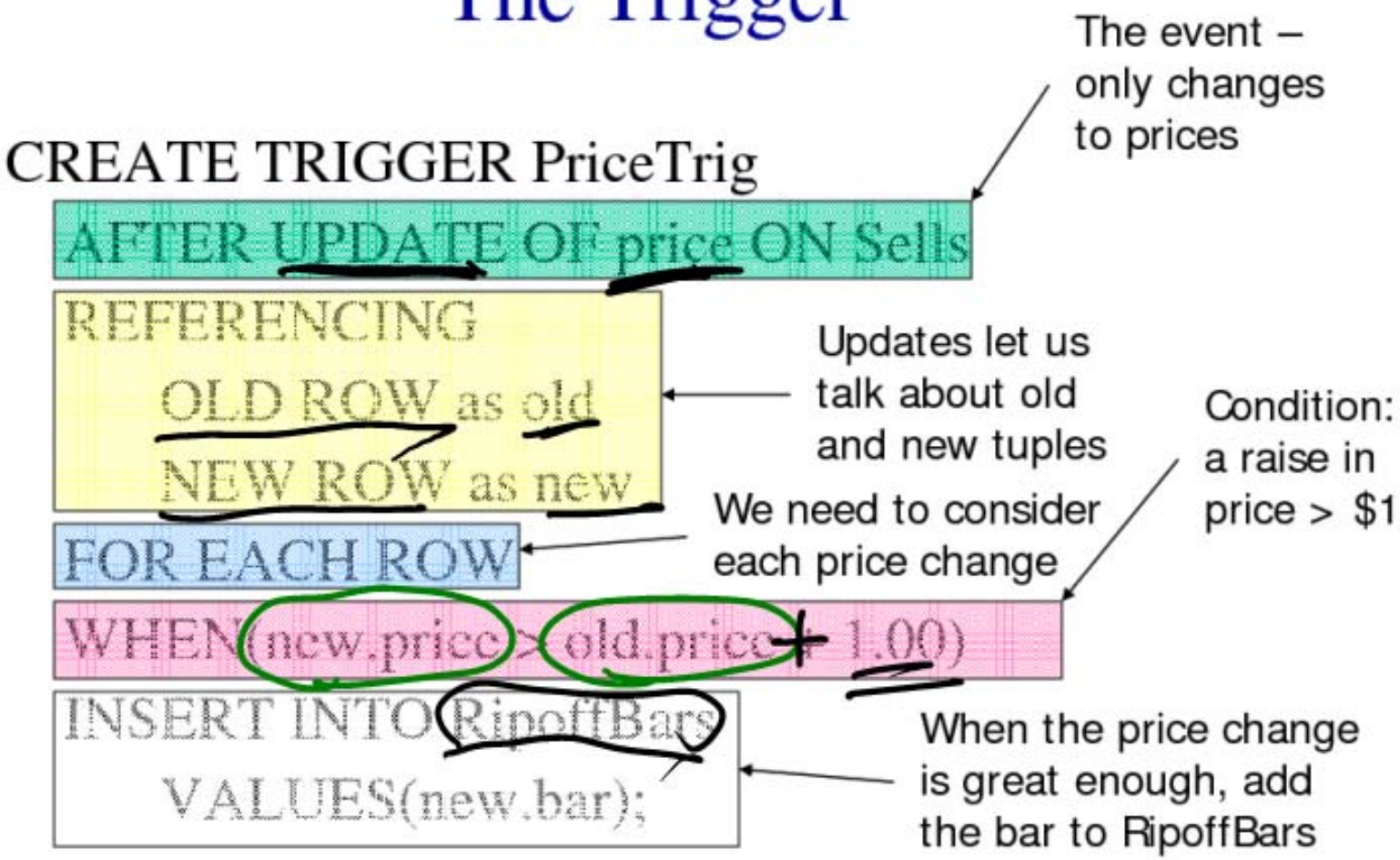
Options: The Action

- There can be more than one SQL statement in the action.
  - Surround by BEGIN . . . END if there is more than one.
- But queries make no sense in an action, so we are really limited to modifications.

Another Example

- Using Sells(bar, beer, price) and a unary relation RipoffBars(bar) created for the purpose, maintain a list of bars that raise the price of any beer by more than $1.

The Trigger


CREATE TRIGGER PriceTrig
AFTER UPDATE OF price ON Sells
REFERENCING
OLD ROW as old
NEW ROW as new
FOR EACH ROW
WHEN(new.price > old.price + 1.00)
INSERT INTO RipoffBars
VALUES(new.bar);
The event – only changes to prices
Updates let us talk about old and new tuples
We need to consider each price change
Condition: a raise in price > $1
When the price change is great enough, add the bar to RipoffBars

Behind the Scene: Why Trigger was invented?

Aspects of a trigger subsystem in an integrated database system. Proceedings of the 2nd international conference on Software engineering. 1976.

1. Extended assertions. (why?)
2. ??

Behind the Scene: This is why…

Aspects of a Trigger Subsystem in an Integrated Database System

by

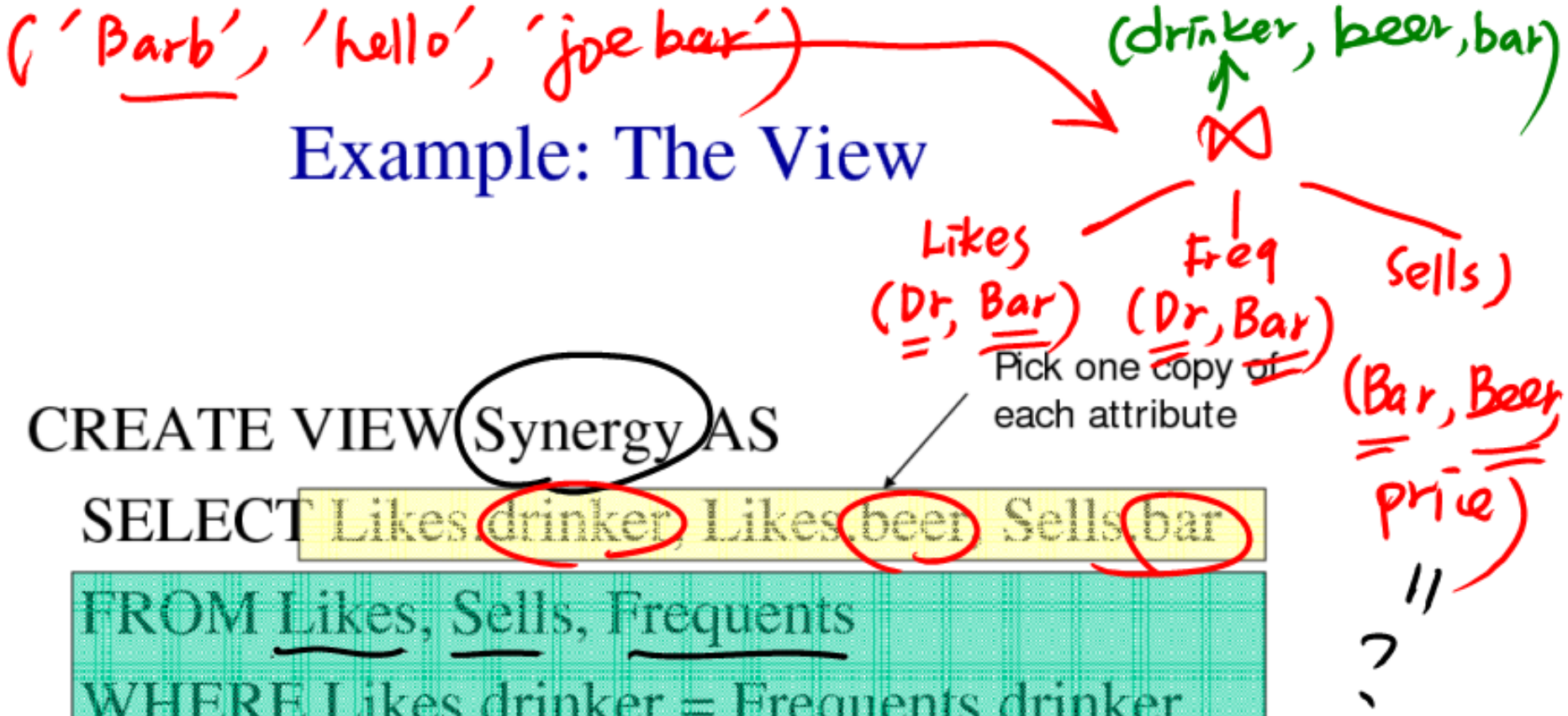
Kapali P. Eswaran
IBM Research Laboratory
San Jose

ABSTPACT. This paper considers the specifications and design of a trigger subsystem in a database management system. The use of triggers as extended assertions and as a means to materialize virtual data objects are discussed. The functional requirements of a trigger subsystem and different implementation issues are studied. We also examine the relationships between a trigger subsystem and the rest of the database system, in particular the authorization and locking subsystems.

Triggers on Views

- Generally, it is impossible to modify a view, because it doesn't exist.
- But an INSTEAD OF trigger lets us interpret view modifications in a way that makes sense.
- Example: We'll design a view Synergy that has (drinker, beer, bar) triples such that the bar serves the beer, the drinker frequents the bar and likes the beer.

Example: The View



Interpreting a View Insertion

- We cannot insert into Synergy --- it is a view.
- But we can use an INSTEAD OF trigger to turn a (drinker, beer, bar) triple into three insertions of projected pairs, one for each of Likes, Sells, and Frequents.
  - The Sells.price will have to be NULL.

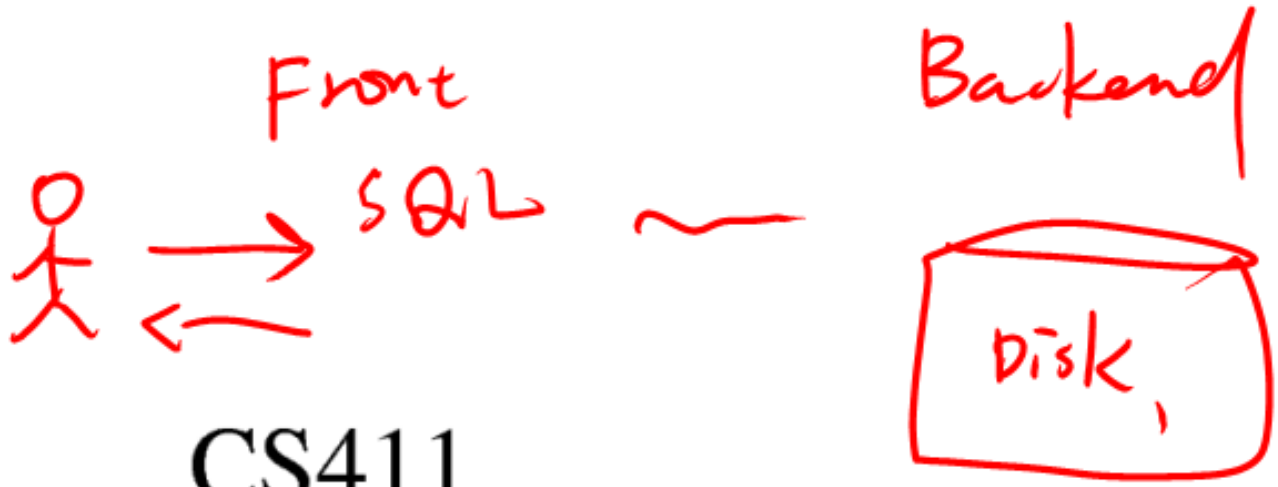
The Trigger

insert into synergy values ('Barb', 'Hello', joe bar)

```
CREATE TRIGGER ViewTrig
INSTEAD OF INSERT ON Synergy
REFERENCING NEW ROW AS n
FOR EACH ROW
BEGIN
    INSERT INTO LIKES VALUES(n.drinker, n.beer);
    INSERT INTO SELLS(bar, beer) VALUES(n.bar, n.beer);
    INSERT INTO FREQUENTS VALUES(n.drinker, n.bar);
END;
```

barb hello

price = Null



CS411
Database Systems

07: Indexing

Announce

~ One week. off.

After that:

~ Special Topics:

on Final { Wed: Anatomy of SQLite.
Fri: SQL Tuning.

Why Do We Learn This?

query does sel/from/where

Find out thing tuples, quickly of course,

Index

by values ⟷ ptr

GPA > 3.0

dept = "CS"

logical declarative

phy procedural

Indexing

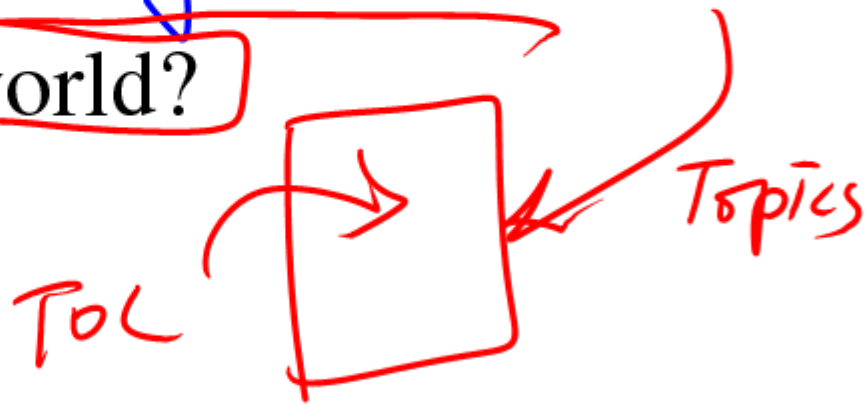
- Indexing
  - types of indexes
  - B+ trees
  - hash tables

Q: What is "indexing"?

- To build an index.
- But what is an index?
- Examples in the real world?

value

an organization of labeled ptrs to collection of items.



What is "indexing"?

Indexes

- An *index* on a file speeds up selections on the *search key field(s)*
- Search key = any subset of the fields of a relation
  - *Search key* is not the same as *key* (minimal set of fields that uniquely identify a record in a relation).
- Entries in an index: (k, r), where:
  - k = the key
  - r = the record OR record id OR record ids

Types of Indexes

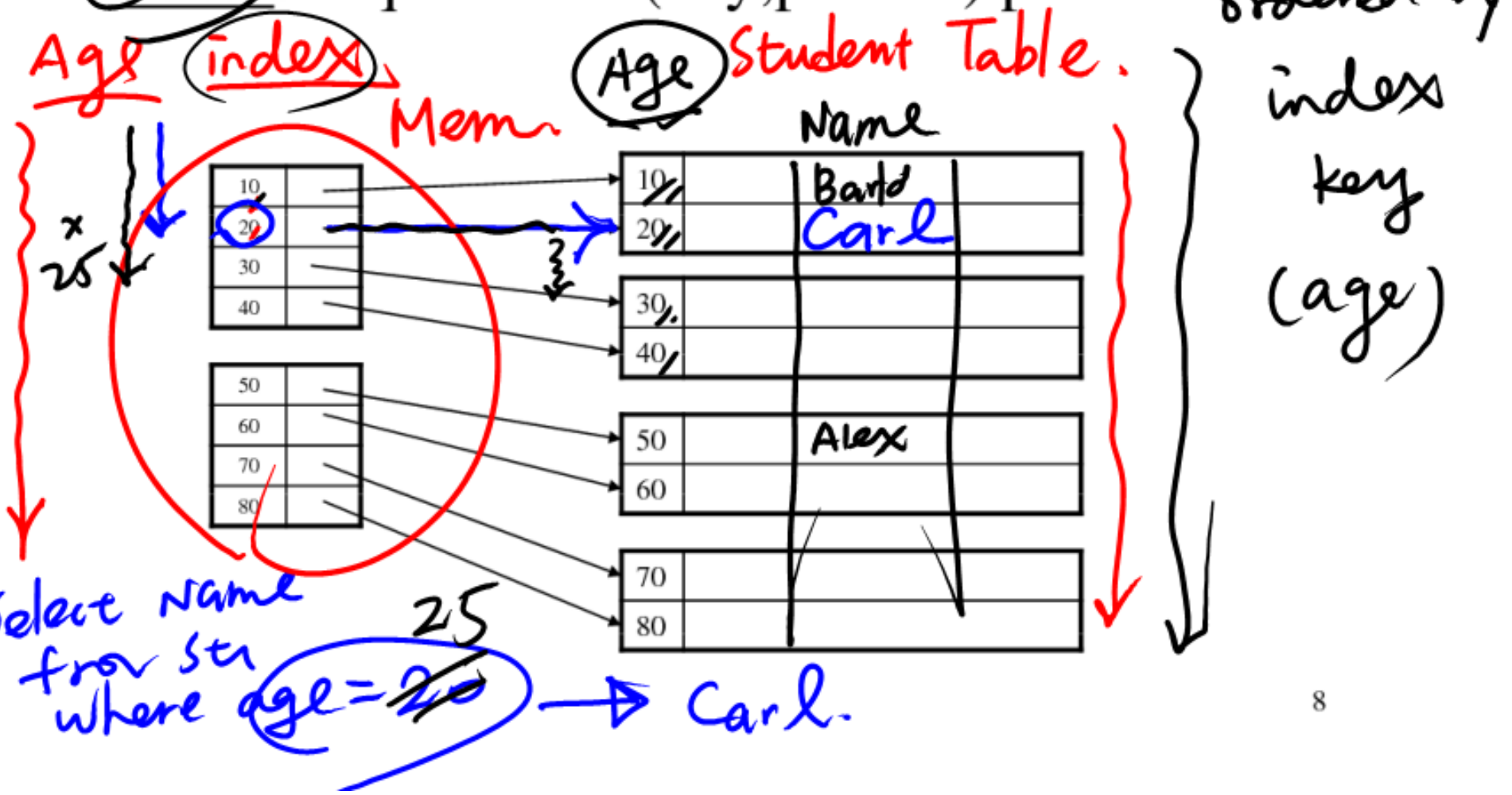
- Clustered/unclustered
  - Clustered = records sorted in the key order
  - Unclustered = no
- Dense/sparse
  - Dense = each record has an entry in the index
  - Sparse = only some records have
- Primary/secondary
  - Primary = on the primary key
  - Secondary = on any key
  - Some textbooks interpret these differently
- B+ tree / Hash table / …

Sequential File before 1972 (B-tree invented)

Ex: Clustered, Dense Index

- Clustered: File is sorted on the index attribute
- *Dense*: sequence of (key,pointer) pairs

every age value appear in Index



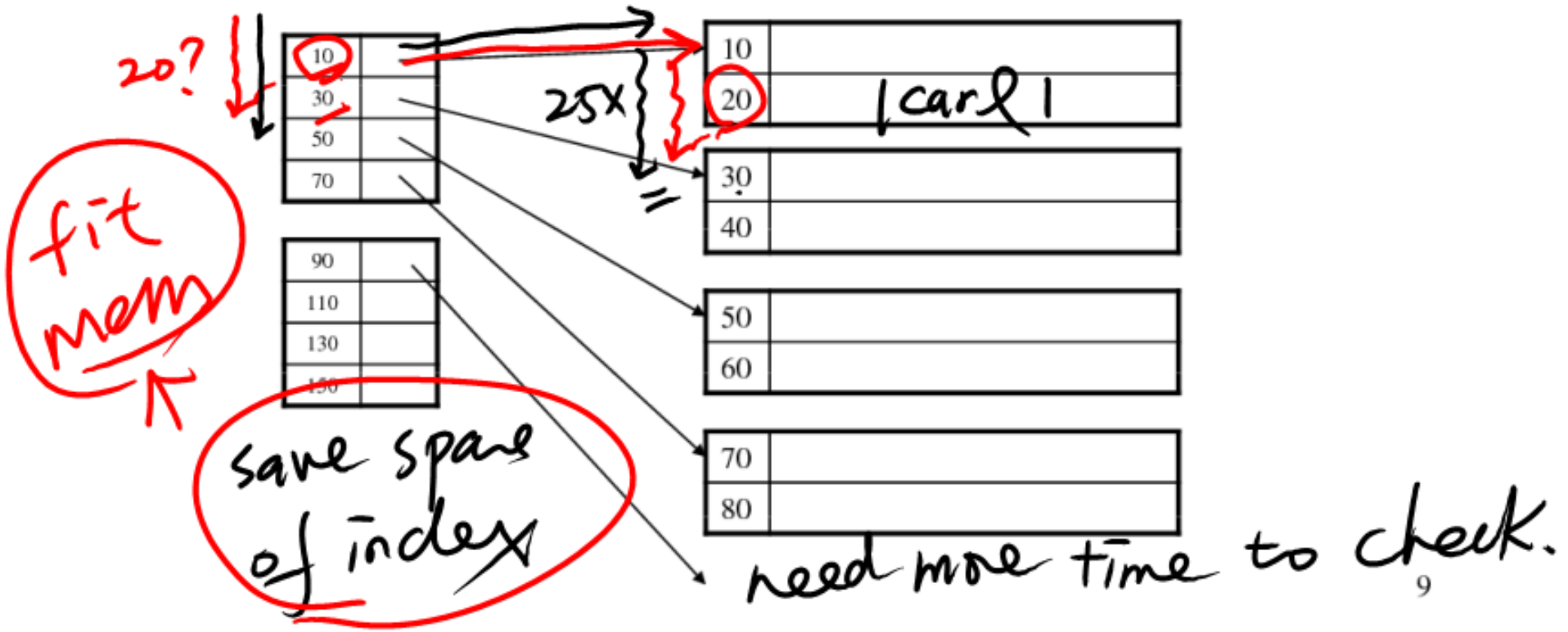
sorted by Age

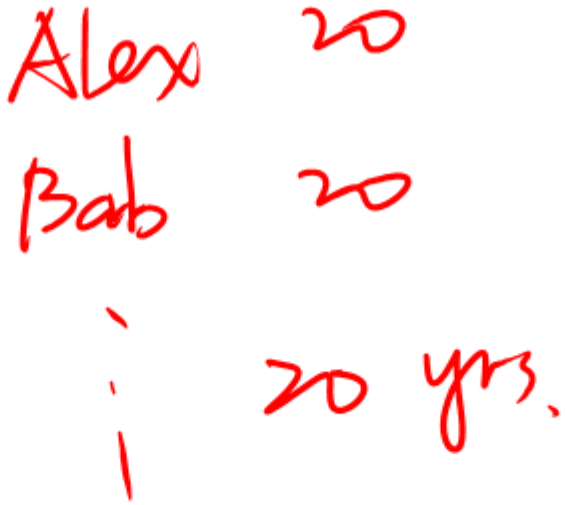
Clustered, Sparse Index

Dense

- *Sparse* index: one key per data block

select * from where age = 25

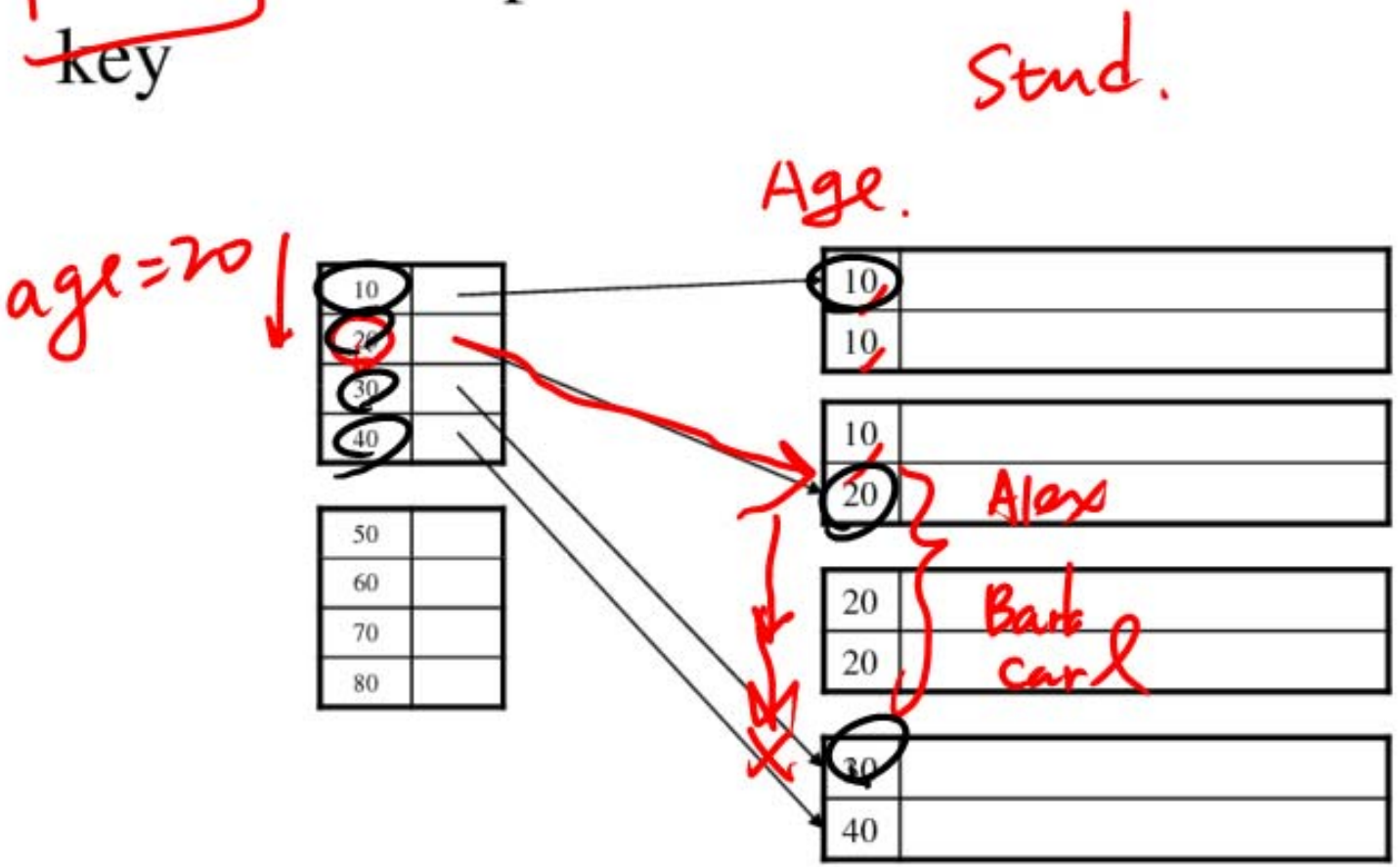




How if duplicate keys?

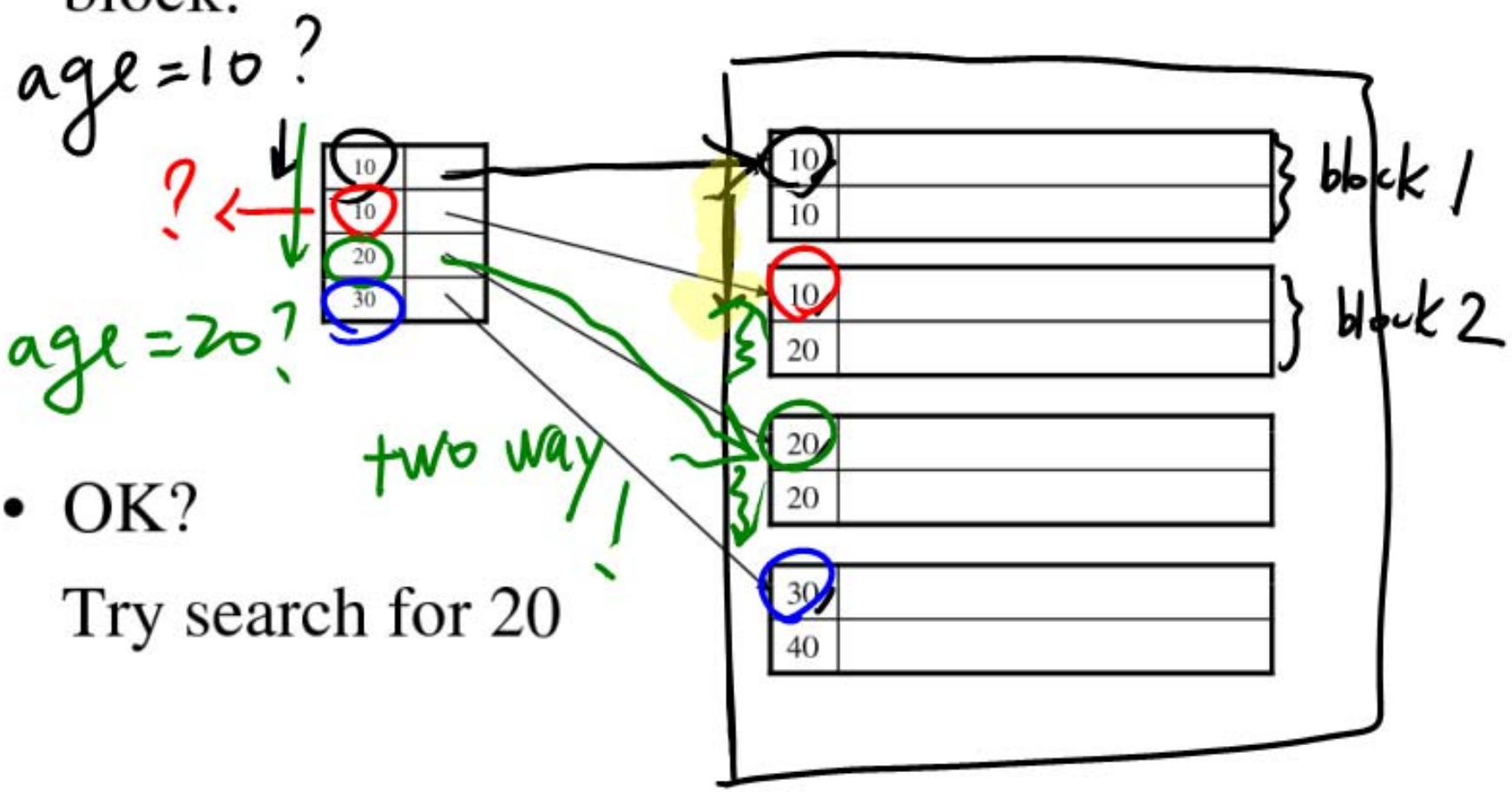
Clustered Index with Duplicate Keys

- Dense index: point to the first record with that key



Clustered Index with Duplicate Keys

- Sparse index: pointer to lowest search key in each block:

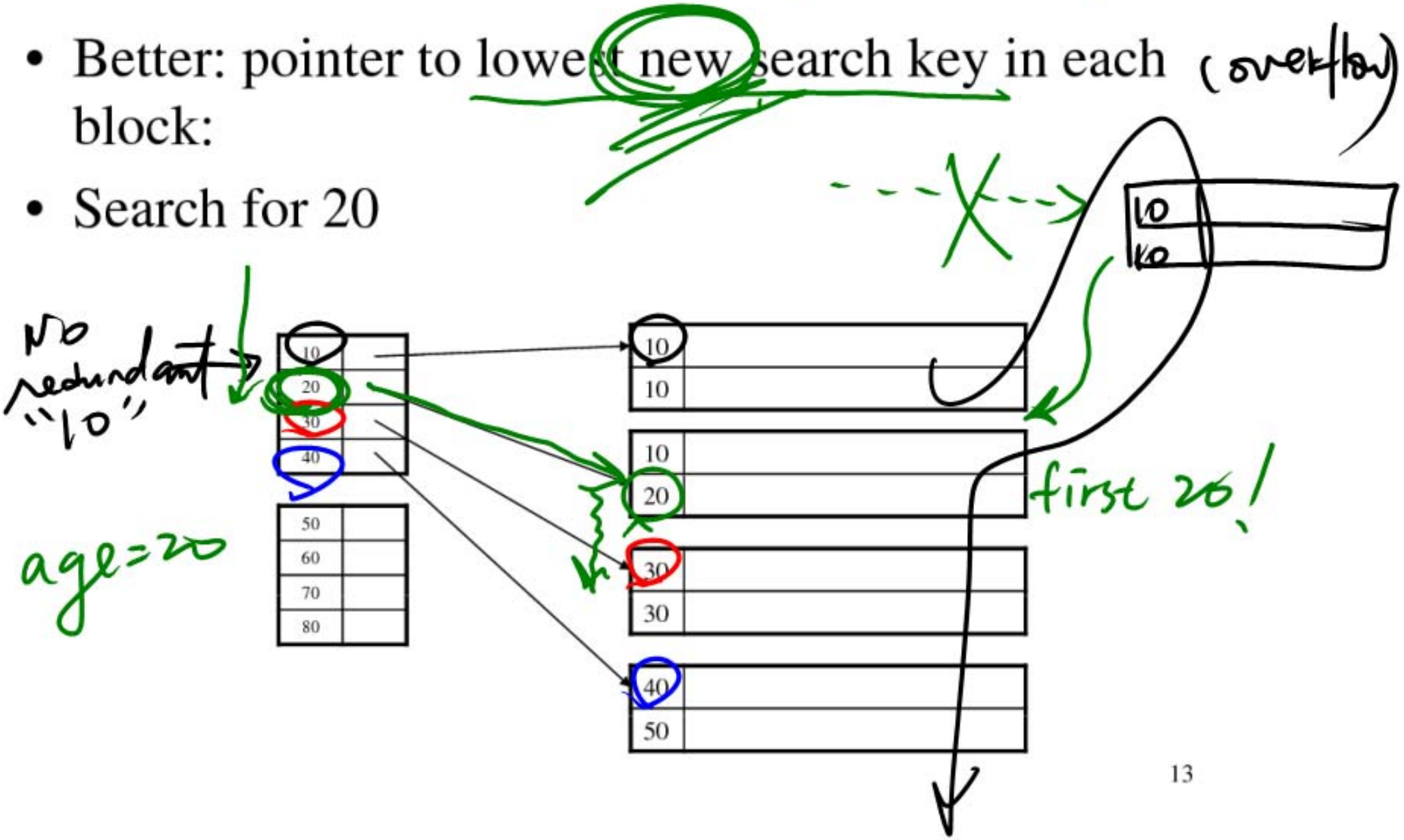


- OK?

  Try search for 20

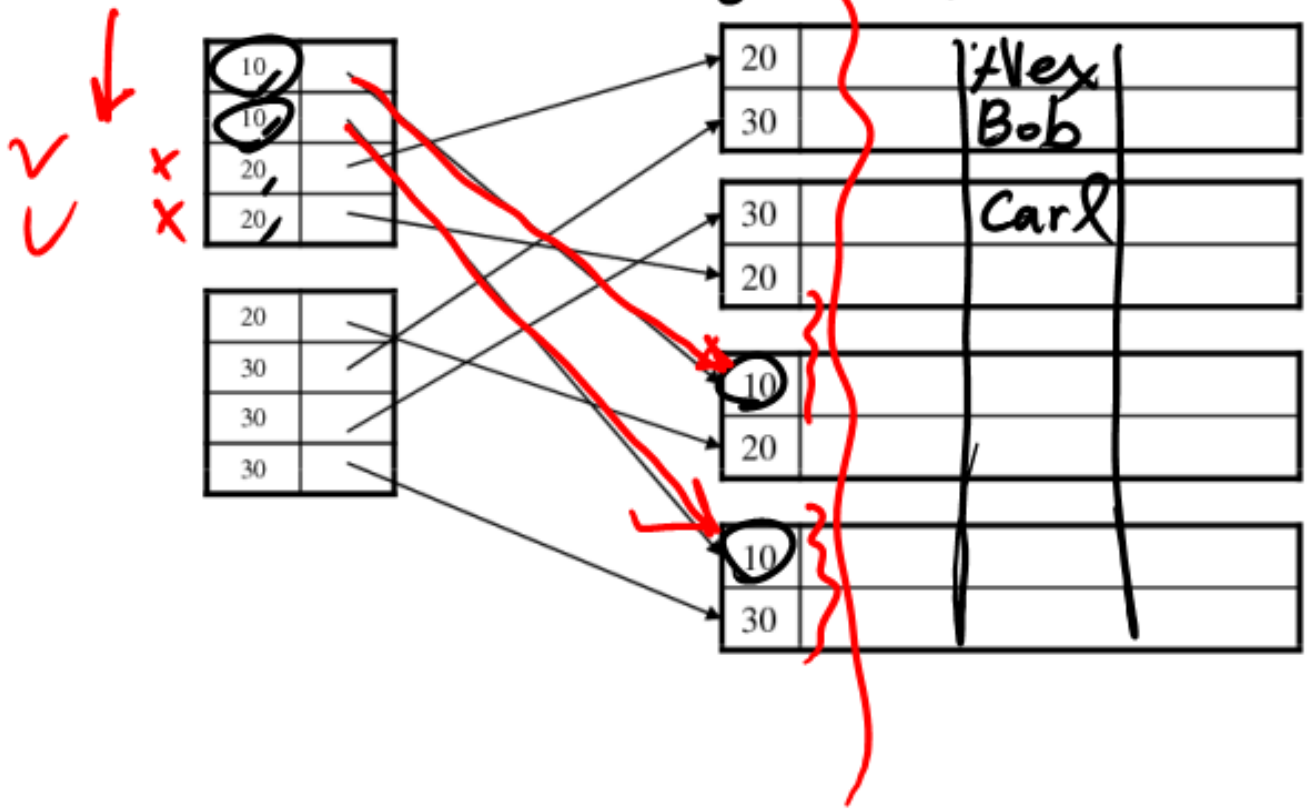
Clustered Index with Duplicate Keys

- Better: pointer to lowest new search key in each block:
- Search for 20

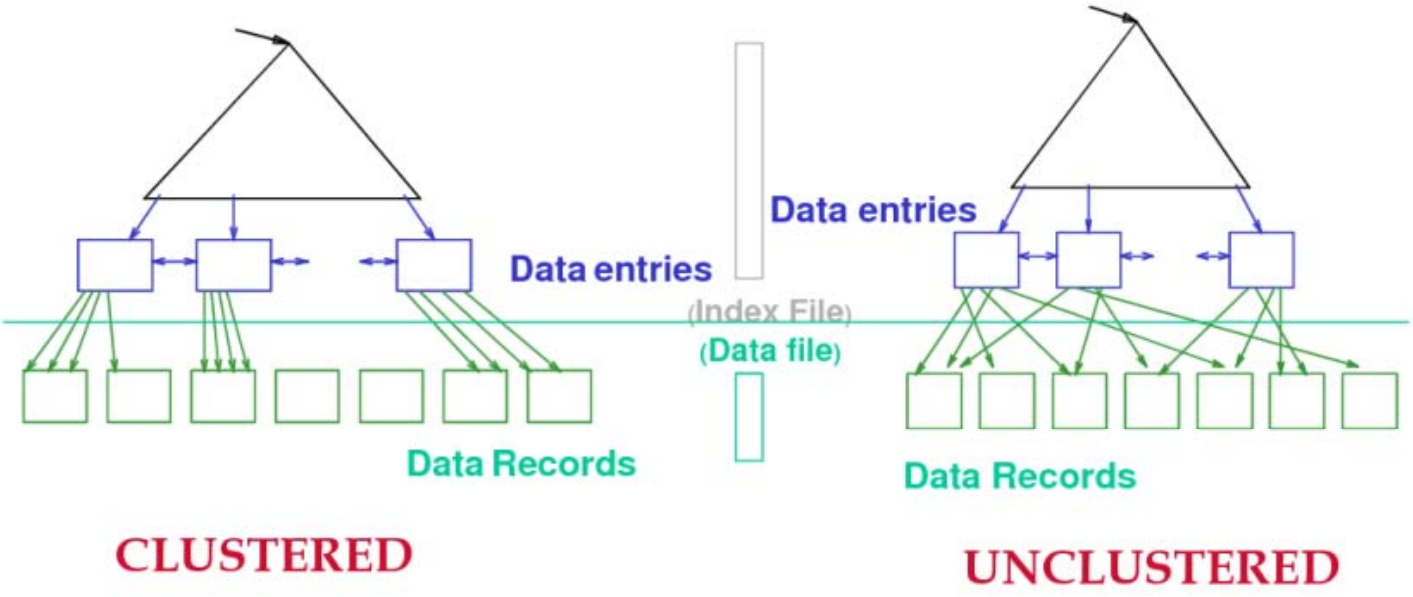


Unclustered Indexes

- Often for indexing other attributes than primary key
- Always dense (why ?)



Summary Clustered vs. Unclustered Index



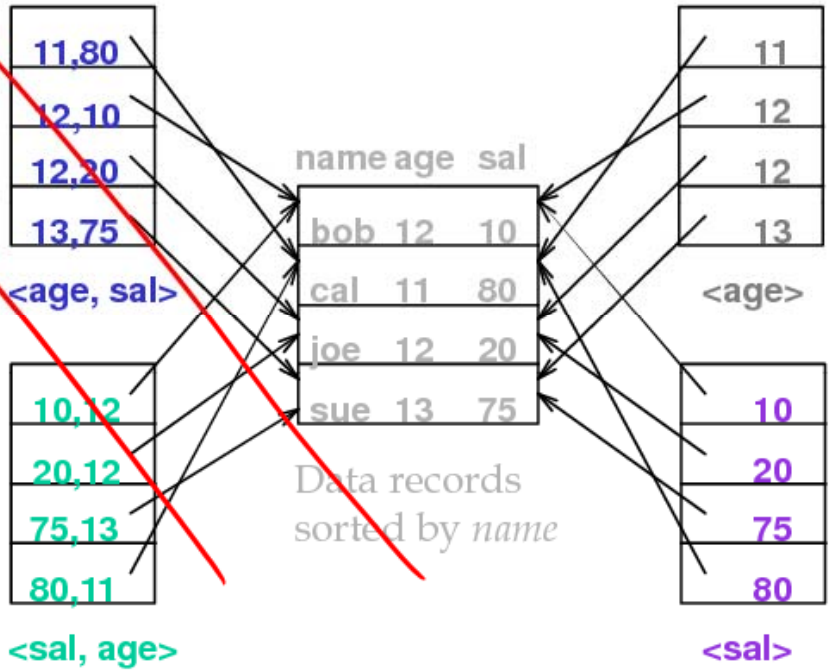
CLUSTERED

UNCLUSTERED

Composite Search Keys

- *Composite Search Keys*: Search on a combination of fields.
  - Equality query: Every field value is equal to a constant value. E.g. wrt <sal,age> index:
    - age=20 and sal =75
  - Range query: Some field value is not a constant. E.g.:
    - age =20; or age=20 and sal > 10

Examples of composite key indexes using lexicographic order.



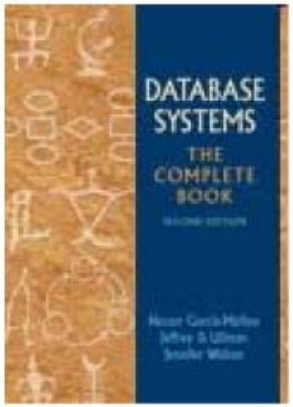
Data entries in index sorted by <*sal,age*>

Data entries sorted by <*sal*>

Q: Our textbook as example: Indexes?

Front Back

2 ToC Topic.



- How many indexes? Where?
- What are keys? What are records?
- Clustered?
- Dense?
- Primary?

	ToC	Topic
key	chap/sec#	topic
records	page	page
clustered	✓	✗
Dense	✓	✓ ← must be dense