[544] SQL Databases (MySQL)

Tyler Caraza-Harter

Outline

Creating/designing tables

- data modeling
- primary/foreign keys

Transactions

Queries

Data Modeling

Data modeling: deciding how to represent something in an underlying system.

Low-level example (protobufs): how will we represent numbers as bytes being sent over a network?

Traditional Databases: how will we represent things/people/events/etc as rows in tables?

option 1:

tbl_orders

name	book	amount	county	state
Tyler Harter	Designing Data-Intensive Applications	23	Dane	WI
Tyler Harter	Learning Spark	38	Dane	WI
Tyler Harter	Cassandra: The Definitive Guide	39	Dane	WI

Keys and Normalization

SQL keys:

- primary key: uniquely identify a row ("id" in tbl_counties)
- foreign key: reference a primary key ("county_id" in tbl_orders)

In database theory we would say option 2 is "more normalized" (note: there are well-defined normalization levels with formal rules -- we won't get into that in 544)

option 1:

tbl_orders

name	book	amount	county	state
Tyler Harter	Designing Data-Intensive Applications	23	Dane	WI
Tyler Harter	Learning Spark	38	Dane	WI
Tyler Harter	Cassandra: The Definitive Guide	39	Dane	WI

option 2:

tbl_orders

name	book	amount	county_id
Tyler Harter	Designing Data-Intensive Applications	23	1
Tyler Harter	Learning Spark	38	1
Tyler Harter	Cassandra: The Definitive Guide	39	1

tbl_counties

id	county	state
1	Dane	WI
2	Milwaukee	WI
3	La Crosse	WI

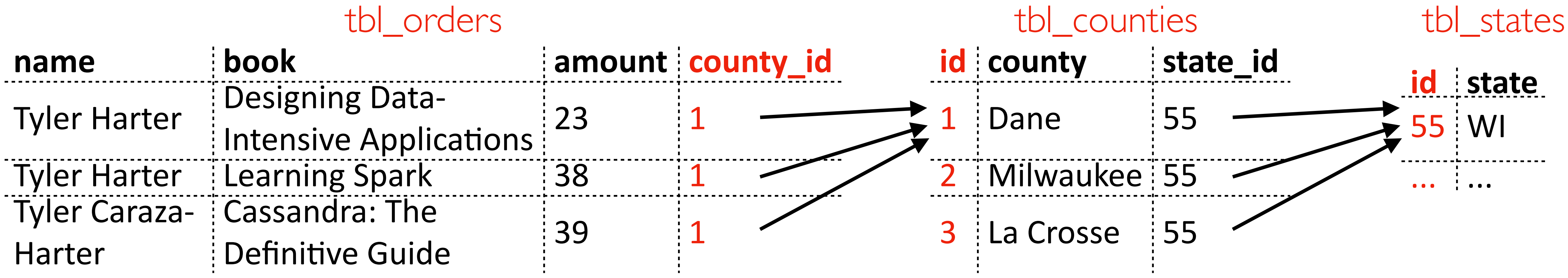
Normalization Tradeoffs

Benefits of more normalization:

- avoid inconsistencies
- changes in the real world correspond to fewer changes in the DB
- often save space

Downsides of more normalization:

- queries are sometimes slower
- historical record keeping (for example, if you need to reproduce an invoice prior to somebody's name change, you might want the name at time of purchase)



tbl_orders

name	book	amount	county_id
Tyler Harter	Designing Data-Intensive Applications	23	1
Tyler Harter	Learning Spark	38	1
Tyler Caraza-Harter	Cassandra: The Definitive Guide	39	1

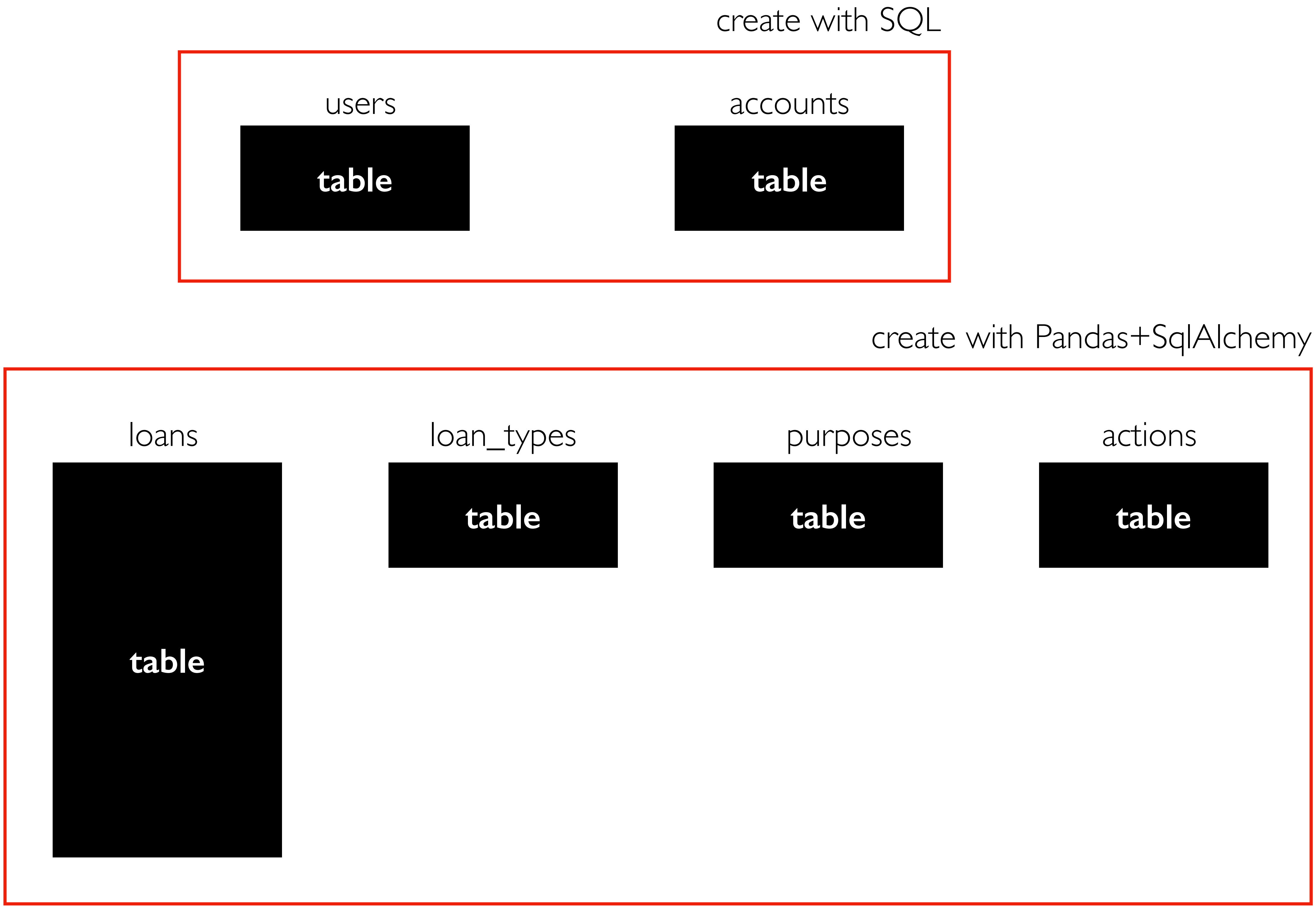
tbl_counties

id	county	state_id
1	Dane	55
2	Milwaukee	55
3	La Crosse	55

tbl_states

id	state
55	WI
...	...

Demos: creating+populating tables



create with SQL

users

table

accounts

table

create with Pandas+SqlAlchemy

loans

table

loan_types

table

purposes

table

actions

table

Outline

Creating/designing tables

- data modeling
- primary/foreign keys

Transactions

Queries

Definitions of Transactions

Definition 1, regarding access patterns

- analytics: calculate over many/all rows, few colums (corresponding DB: OLAP)
- transactions: work with whole row or few rows at a time (corresponding DB: OLTP)

Definition 2, regarding guarantees for a collection of DB operations (often changes). Common guarantees:

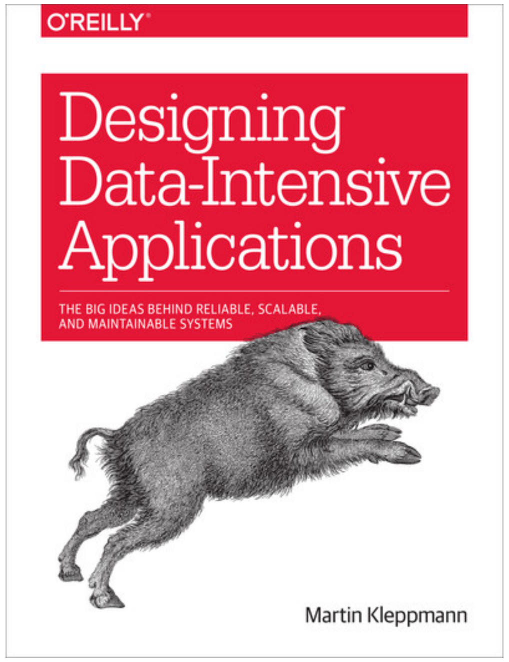
- atomicity: it all happens or nothing happens (partial progress is rolled back)
- consistency: application invariants (like no negative bank accounts) are supported
- isolation: others cannot see a transaction in progress (aka *atomicity* when talking about locks)
- durability: once finished, it persists (even if machine crashes+restarts)

Transactions in a DB are similar to critical sections in a multi-threaded process:

```
if bank_accounts[src] >= dollars:
    bank_accounts[src] -= dollars
    bank_accounts[dst] += dollars
```

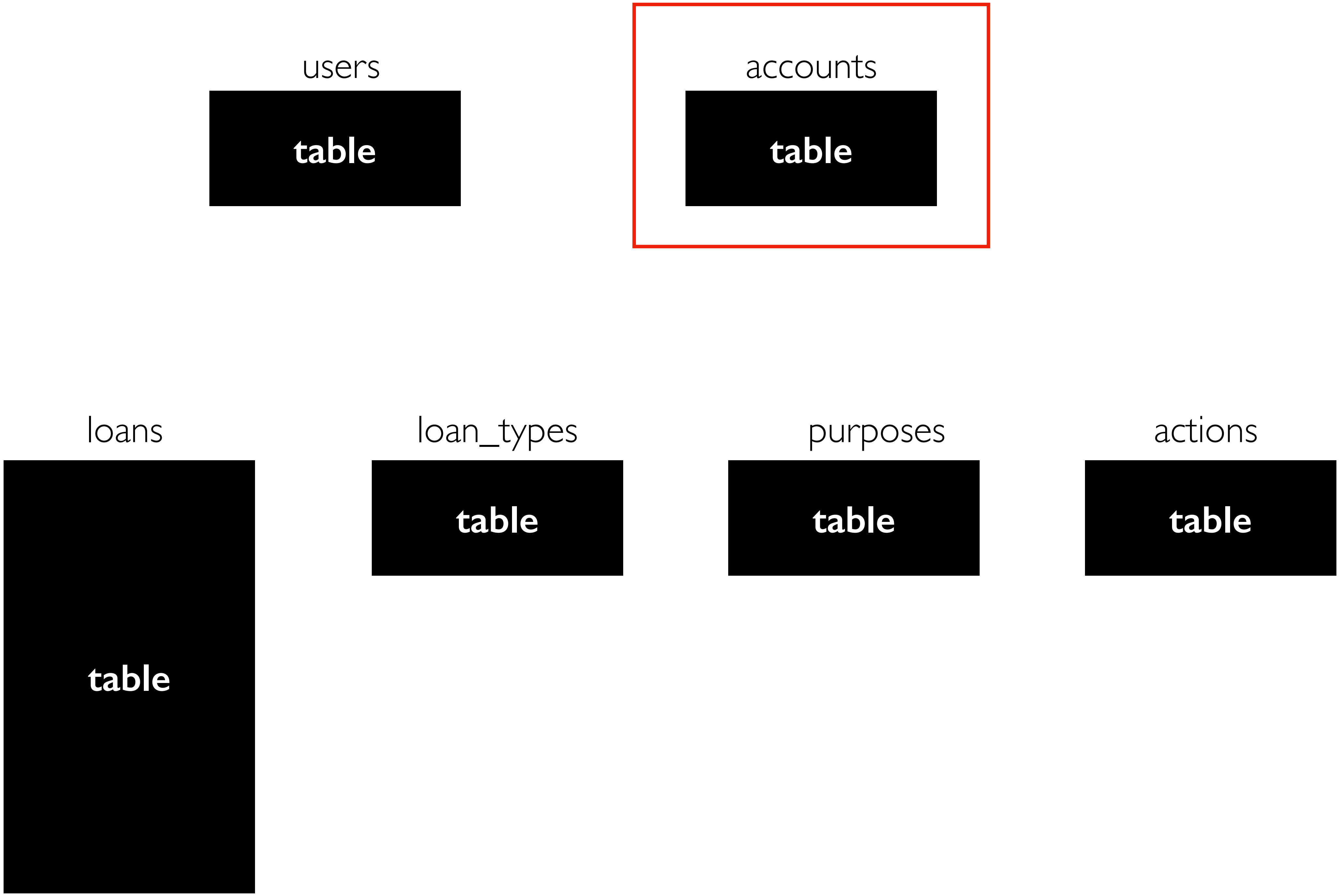
critical section

(example from "locks" lecture)


O'REILLY
Designing Data-Intensive Applications
THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS
Martin Kleppmann

"The Meaning of ACID"

Demo: transactional banking transfer



users

table

accounts

table

loans

table

loan_types

table

purposes

table

actions

table

Outline

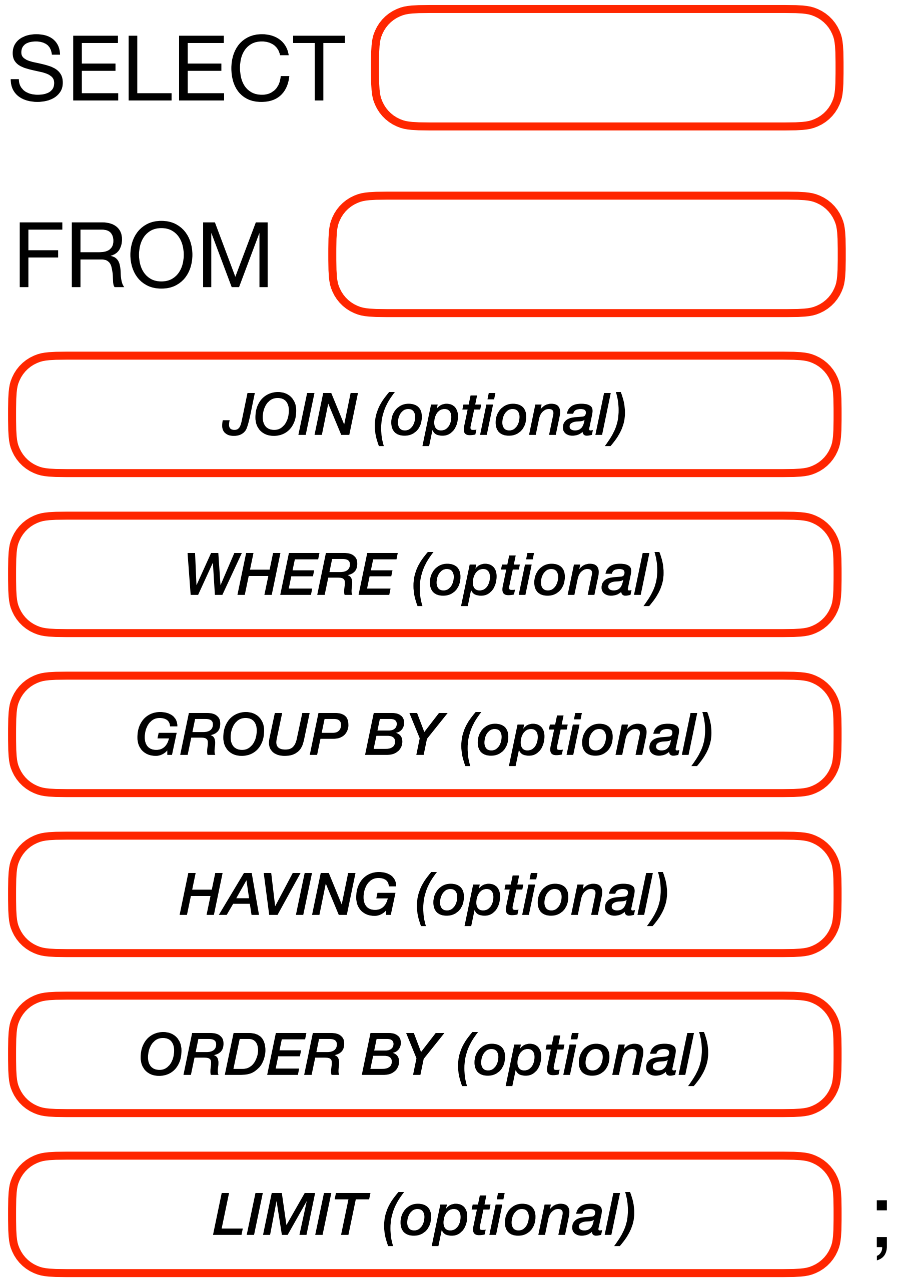
Creating/designing tables

- data modeling
- primary/foreign keys

Transactions

Queries

SQL Query: General Structure



SELECT

FROM

JOIN (optional)

WHERE (optional)

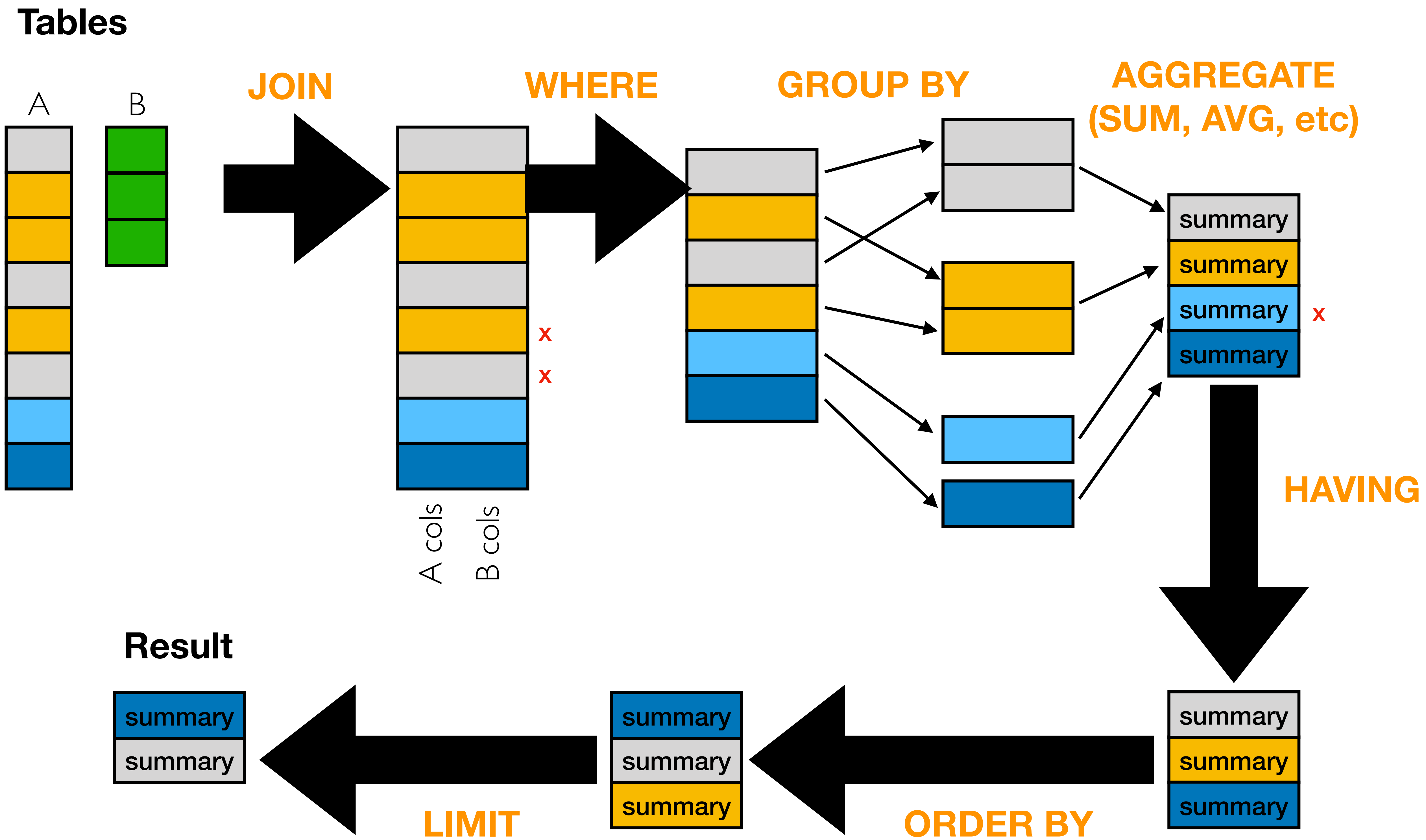
GROUP BY (optional)

HAVING (optional)

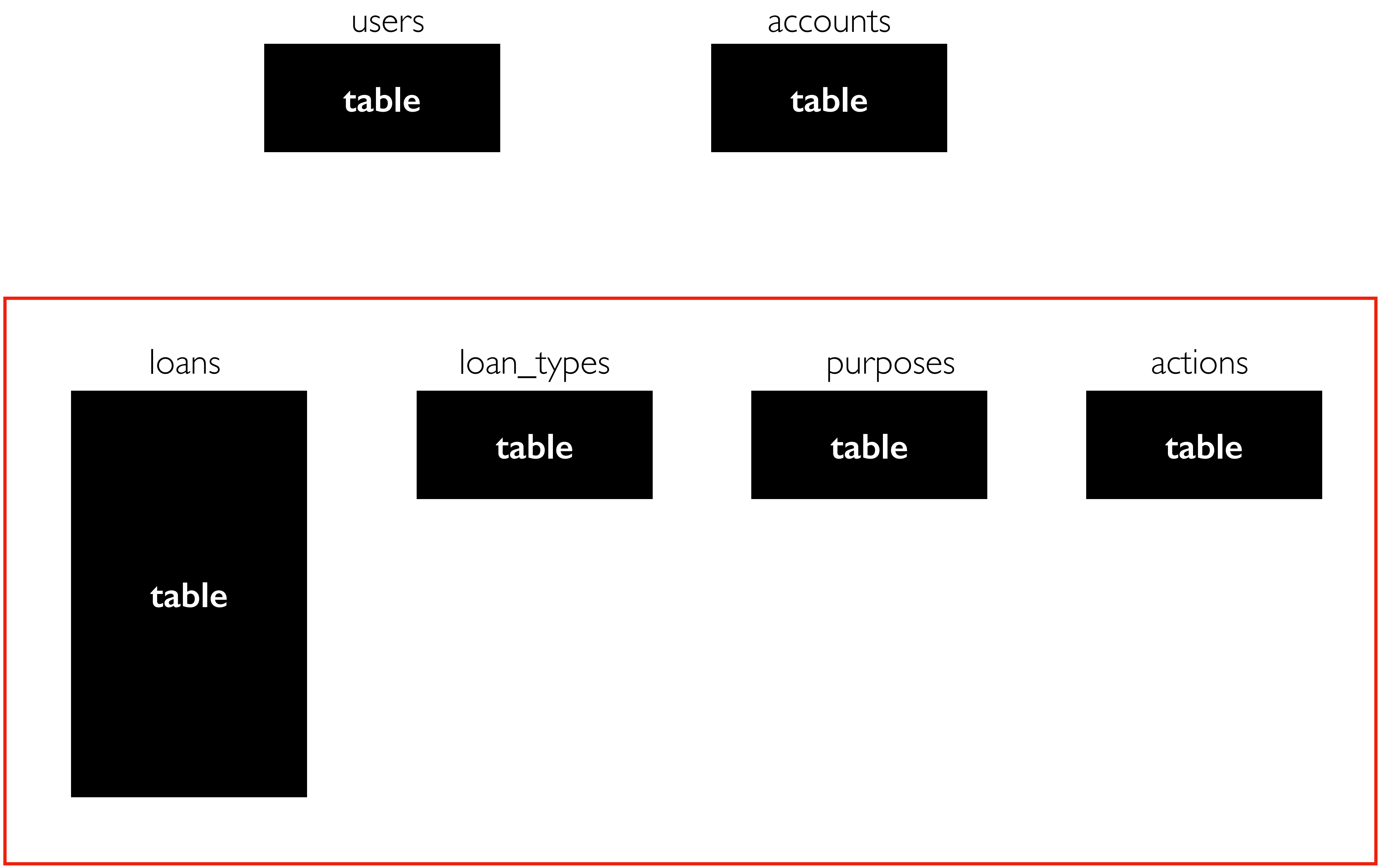
ORDER BY (optional)

LIMIT (optional) ;

Query: a series of transformations



Demos: answering questions about loans



users

table

accounts

table

loans

table

loan_types

table

purposes

table

actions

table