

# Relational Lenses: A language for defining updateable views

Aaron Bohannon Jeffrey A. Vaughan Benjamin C. Pierce

## Key Idea

Every expression in the language denotes a bidirectional transformation called a lens (a view definition together with a view update policy).

## The View Definition Language

- The sources and targets of lenses are database states (sets of named relations with associated schemas).
- Schemas include predicates and functional dependencies. Both play a significant role in determining view update policies.
- A small set of basic lenses is provided.
  - Each basic lens corresponds to a simple relational operation.
  - Additional parameters determine view update policies.
- A composite lens expression can be read from left to right to describe a composite view definition and from right to left to describe a composite view update policy.

### A Composite Lens:

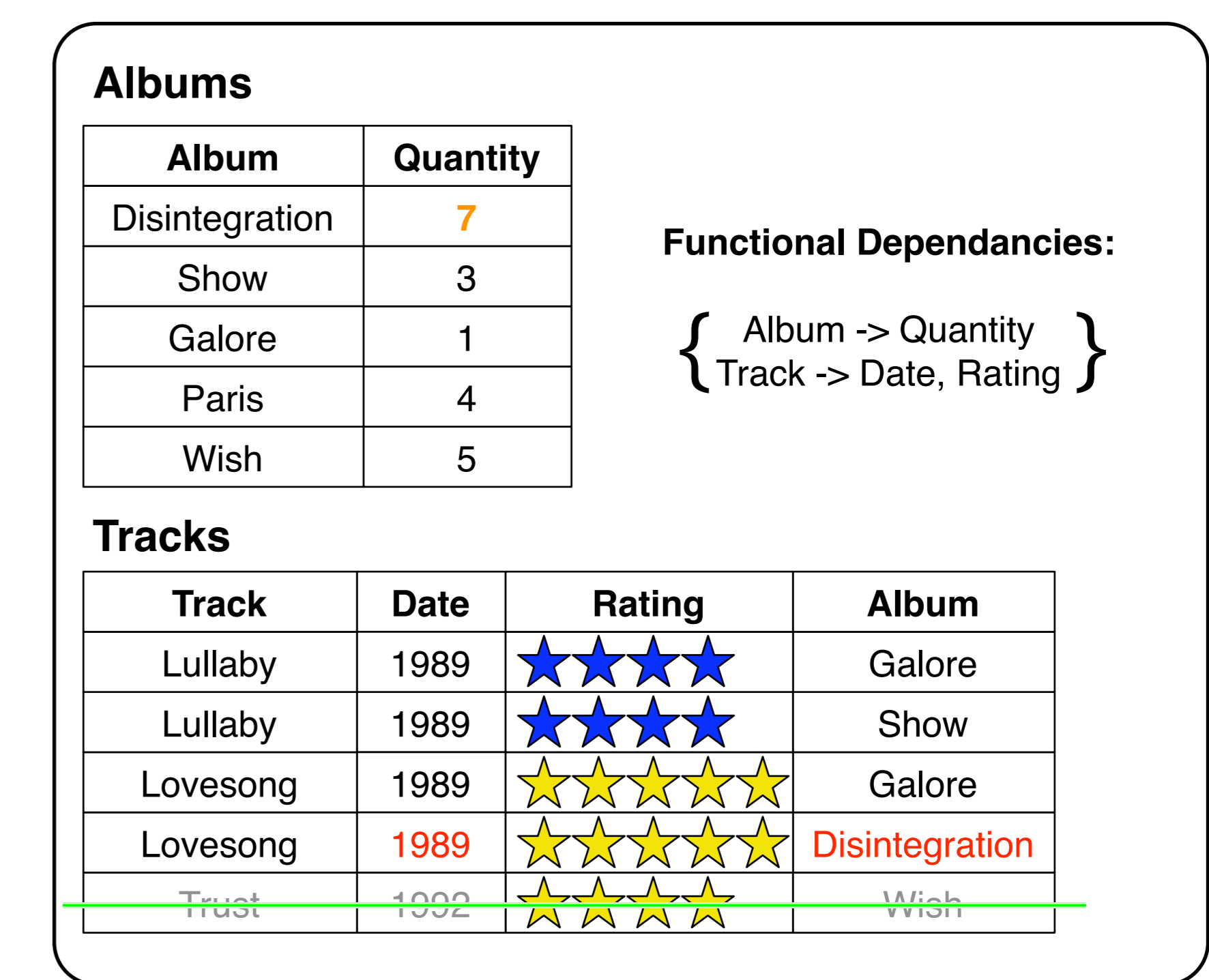
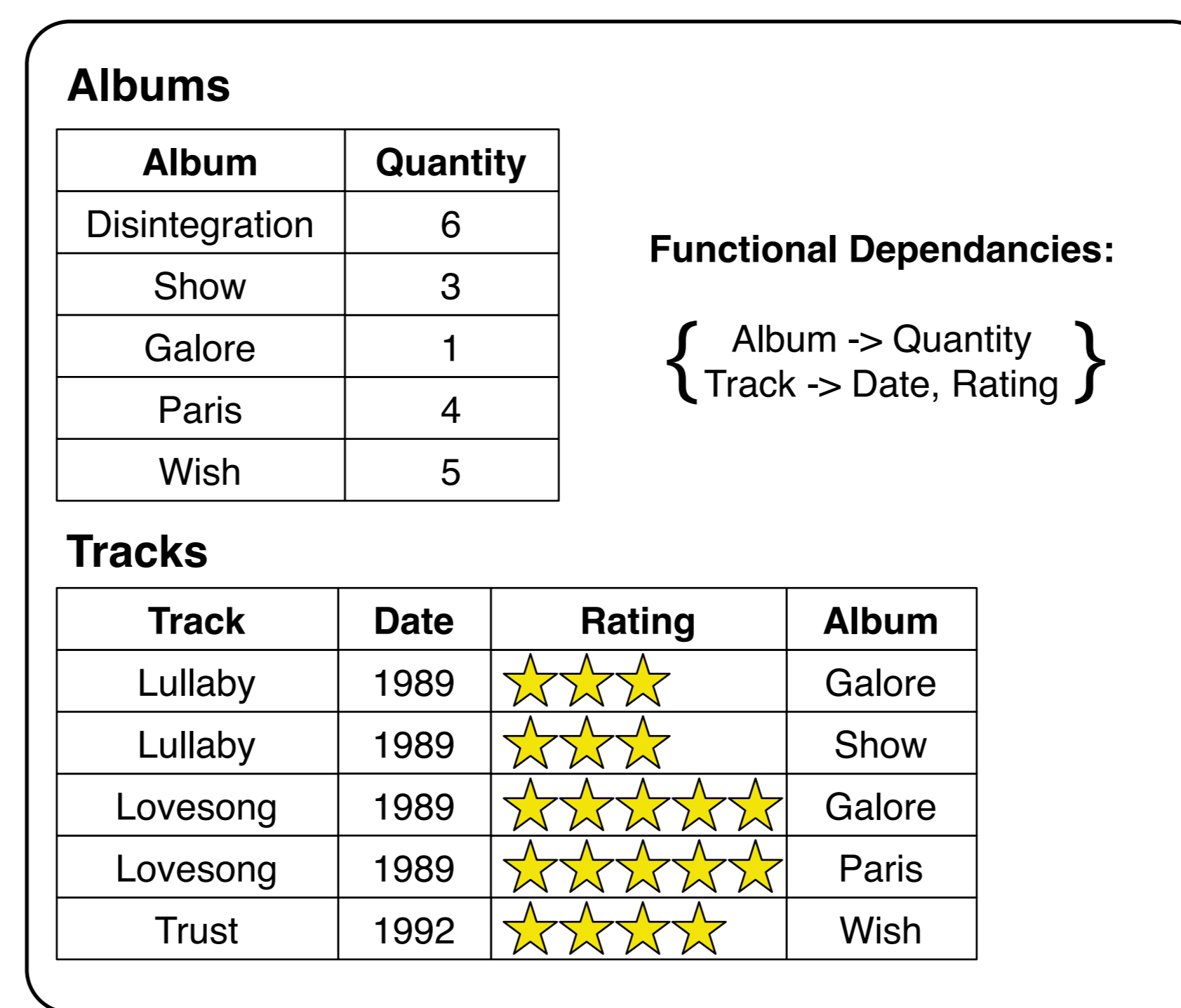
*join Tracks, Albums deleting from Tracks;*  
*project Tracks on Track, Rating, Album,*  
*Quantity with default {Date=Unknown};*  
*select from Tracks where Quantity > 2*

## Static Checking

- We are interested in lenses that are well behaved:
  - View definition and update policy "fit together" in a suitable sense.
  - Any consistent updated view state is mapped to a consistent updated database state (totality).
- Well-behavedness is guaranteed by static checking.
  - Each primitive lens comes with a typing constraint guaranteeing its well-behavedness.
  - Well-behavedness of well-typed composite lenses follows by construction.

## Key Research Challenge

Detailed design of the basic lenses and their typing constraints.



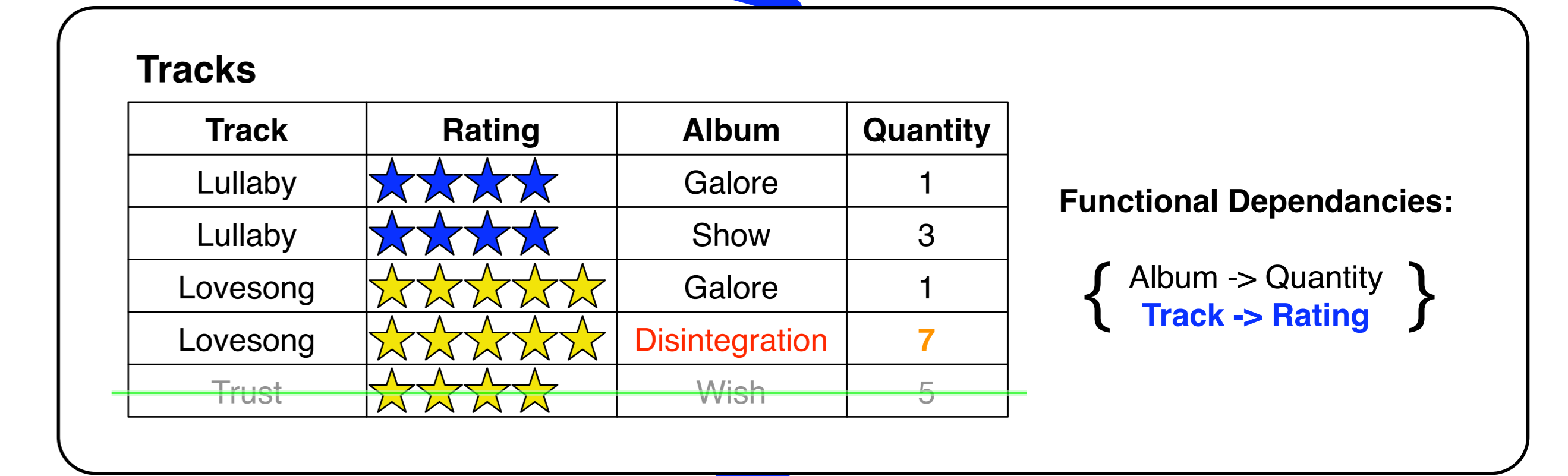
*join Tracks, Albums deleting from Tracks*

In this example, the missing row can be realized by a removal from either Albums or Tracks. The lens expression defining join indicates that rows from Tracks should be deleted when such ambiguous situations arise during an update.



*project Tracks on Track, Rating, Album, Quantity with default {Date=Unknown}*

In project's reverse direction, missing information can be recovered from the database state or supplied as a default. Here, the year 1989 is inferred for the third row using the Track -> Date functional dependency.



*select from Tracks where Quantity > 2*

Select's update policy specifies that it changes existing records to agree with new ones when necessary to satisfy functional dependencies. Note the first row's updated Rating field.



**An update consistent with the view schema.**  
 Rating of Lullaby becomes 4  
 Lovesong's album becomes Disintegration  
 Trust deleted

Views are edited offline: a single "edit" yields an arbitrary new state.

