CompSci 590.6
Understanding Data:
Theory and Applications

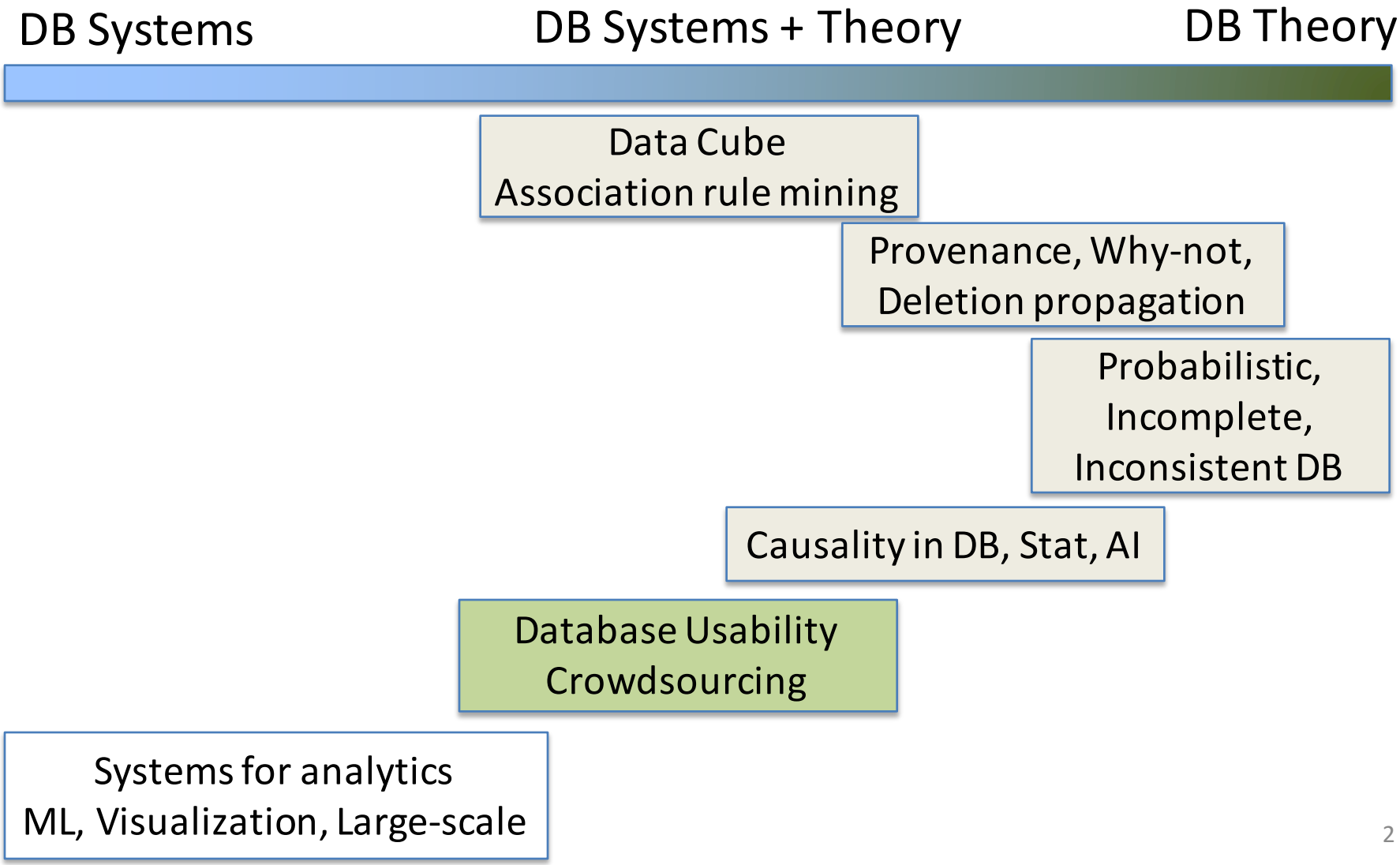
Lecture 18

Database Usability

Instructor: Sudeepa Roy
Email: *sudeepa@cs.duke.edu*

Fall 2015

What did we learn so far?
What will we learn?



Today's Reading

Main reading:

Jagadish-Chapman-Elkiss-Jayapandian-Li-Nandi-Yu

SIGMOD 2007

Making Database Systems Usable

(Student Presentation)

Additional reading:

Li-Chan-Maier

VLDB 2015

Query From Examples: An Iterative, Data-Driven Approach to Query Construction

(An overview in these slides)

Query By Examples (QFE)

- Help database users unfamiliar with SQL construct SQL queries
- User gets (D, R) pair as input
  - D = input database, R = desired result set
- Many such candidate Qs
  - Asks the user to distinguish them again with examples
  - Only requires that the user is able to determine whether a candidate is the result of her intended query on some database D'
- Objective: minimize the effort needed by the user

Example

EXAMPLE 1.1. *To illustrate our QFE approach, suppose that a user needs help to determine her target query Q for the following database-result pair (D, R), where D consists of a single table.*

Employee

Eid	name	gender	dept	salary
1	Alice	F	Sales	3700
2	Bob	M	IT	4200
3	Celina	F	Service	3000
4	Darren	M	IT	5000

Database D

name
Bob
Darren

Result R

For simplicity, assume that there is a set of three candidate queries, $QC = \{Q_1, Q_2, Q_3\}$, for Q, where each $Q_i = \pi_{name}(\sigma_{p_i}(Employee))$, with p_1 = 'gender = "M"', p_2 = 'salary > 4000', and p_3 = 'dept = "IT"'. To help identify the user's target query among these three candidates, our approach will first present to the user a modified database[1] D_1 and two possible query results, R_1 and R_2, on D_1:

Employee

Eid	name	gender	dept	salary
1	Alice	F	Sales	3700
2	Bob	M	IT	3900
3	Celina	F	Service	3000
4	Darren	M	IT	5000

Database D_1

name
Bob
Darren

Result R_1

name
Darren

Result R_2

The modified database D_1 serves to partition QC into multiple subsets. In this example, QC is partitioned into two subsets with the queries in $\{Q_1, Q_3\}$ producing the same result R_1 on D_1 and the query in $\{Q_2\}$ producing the result R_2. The user is then prompted to provide feedback on which of R_1 and R_2 is the result of her target query Q on D_1. If the user chooses R_2, then we conclude that the target query is Q_2. Otherwise, $Q \in \{Q_1, Q_3\}$ and the feedback process will iterate another round and present the user with another modified database D_2 and two possible results, R_3 and R_4 on D_2:

Employee

Eid	name	gender	dept	salary
1	Alice	F	Sales	3700
2	Bob	M	Service	4200
3	Celina	F	Service	3000
4	Darren	M	IT	5000

Database D_2

name
Bob
Darren

Result R_3

name
Darren

Result R_4

If the user feed back that R_3 is the result of Q on D_2, then we conclude that Q is Q_1; otherwise, we conclude that Q is Q_3. For this example, the target query is determined with at most two rounds of user feedback, each of which involves a single change in the database. □

QFE : Challenges

1. How to generate candidate target queries given an initial database-result pair
   - Not the focus of this paper
   - Tran-Chan-Parthasarathy:"Query by Output" (SIGMOD 2009)
   - Zhang-Elmeleegy-Procopiuc-Srivastava: "Reverse engineering complex join queries" (SIGMOD 2013)

2. How to optimize the user-feedback interactions to minimize the user's effort to identify the desired query
   - This paper
   - Select-Project-Join queries

Architecture and Execution

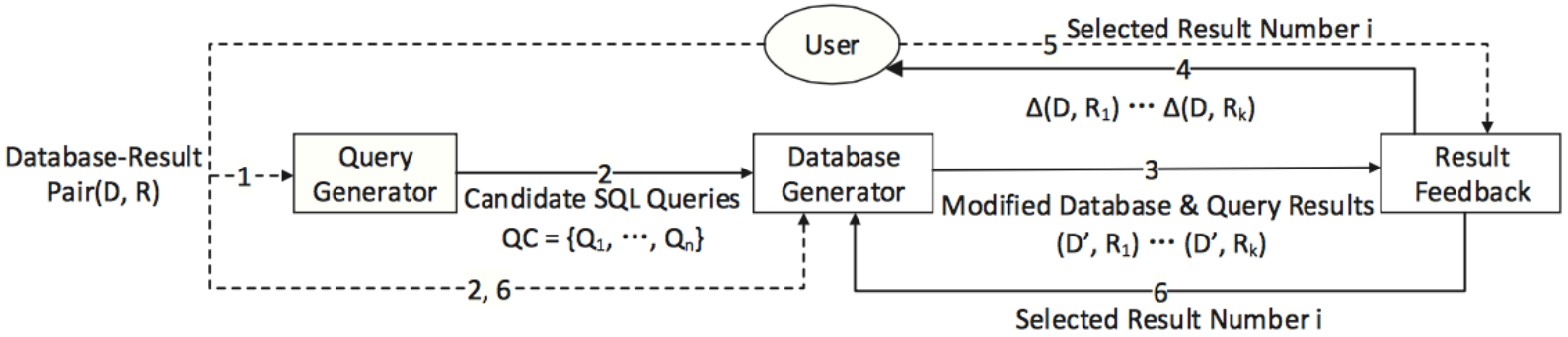


Figure 1: Overall Architecture of QFE

Architecture and Execution

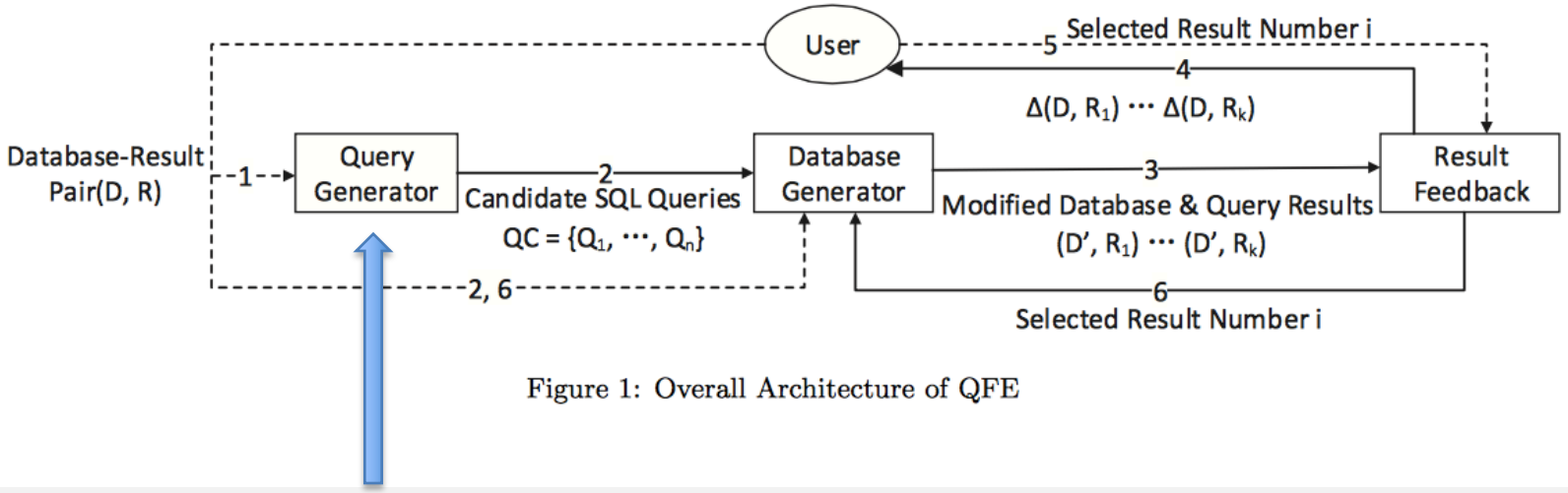


Figure 1: Overall Architecture of QFE

The Query Generator module

- takes (D,R) as input
- generates a set of candidate SQL queries QC = $\{Q_1, \cdots, Q_n\}$ for (D,R)
  - i.e., $Q_i(D) = R$ for each $Q_i \in QC$

Overview: Query Generator

- Tree-based classifier
  - Positive tuples: contribute to query result
  - Negative tuples: do not contribute
- A binary decision tree is constructed top-down
  - If a leaf-node is not good, split it
  - goodness condition: entropy, classification error, Gini index
  - split with some condition: e.g. t.A <= v

Architecture and Execution

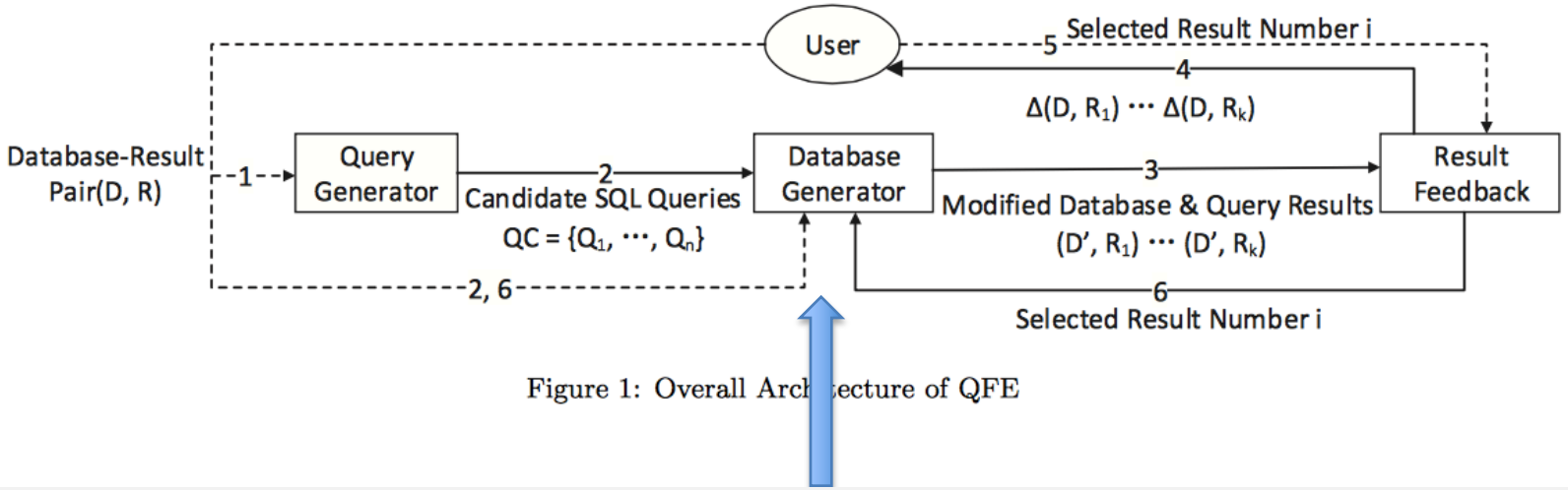


Figure 1: Overall Architecture of QFE

The Database Generator module

- takes (D,R) and QC' $\subseteq$ QC as input
- generates a new database D'
- D' partitions QC' based on their results into k smaller subsets
  - query in the same partition produces the same result

Architecture and Execution

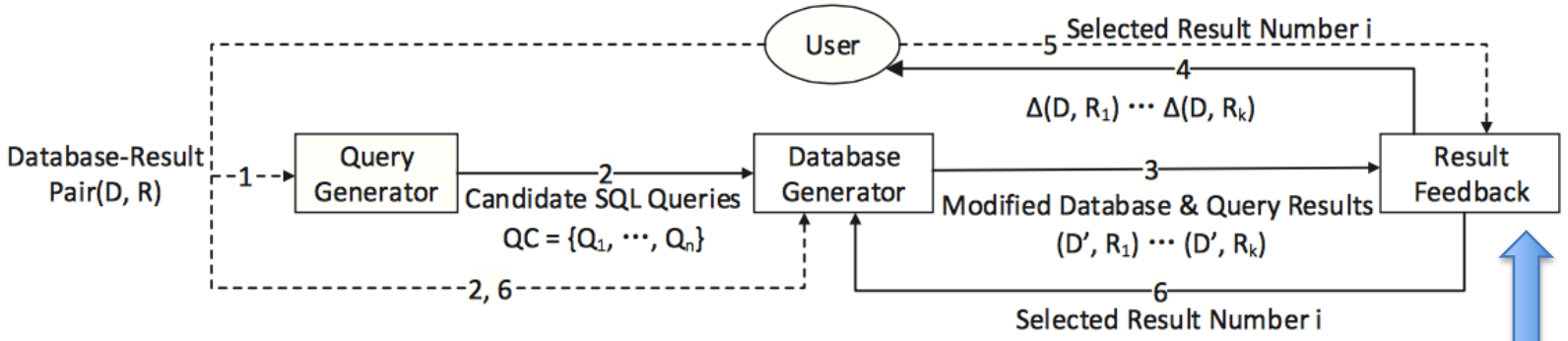


Figure 1: Overall Architecture of QFE

The Result Feedback module

- takes the new database D' and the k results (from k partitions)
- User identifies one partition x as correct
- Repeat with this partition until the chosen partition has only one query

- To help reduce user's effort, only the difference of D' with the original database D is presented.

Cost Model

- Used by the "Database Generator" module to select a "good" modified database D' to partition the query candidates QC into QC_1, …, QC_k
- To minimize the #iterations, each partition should ideally be balanced
  - Remember O(n log n)-time divide and conquer algorithms
- To reduce user's effort
  - D' should be close to D
  - New results R_1,…,R_k should be close to original result R

Balance Score

- Candidate query groups $C = \{QC_1, \ldots, QC_k\}$
- The balance score of D' is σ/k
  - σ = standard deviation of $|QC_1|, \ldots, |QC_k|$
- Smaller balance score

  = many subsets of about the same size

Estimating User's Effort

- Minimize distances between (databases D and D') or (results R_1,..R_k and R)

- Cost components for identifying differences:

1. Current cost
   A. Databases D and D'
      Edit Distance between D and D' minEdit(D, D')
      + Cost proportional to #modified relations

   B. Results R_i and R for i = 1..k
      Sum of edit distances between R_i and R
2. Residual cost
   A. An estimate of the cost for future rounds
   B. Depends on user's feedback
   C. Conservative estimate of #iterations x current cost in each iteration
      Two partitions
      Largest group is chosen

- Large search space – difficult to find D with min cost(D')

Tuple Class: Partitioning Attribute Domain

- Need to find equivalent query classes
- Given a set of queries QC
  - Partition the domain of an attribute A into minimum collection of disjoint subsets $P_{QC}(A)$
  - such that for every subset I and for each selection predicate p on A in QC
  - either every value in I satisfies p or no value in I satisfies p

EXAMPLE 5.1. *Consider a relation* $T(A, B, C)$ *where both* A *and* B *have numeric domains; and a set of queries* $QC = \{Q_1, Q_2\}$, *where* $Q_1 = \sigma_{(A \leq 50) \wedge (B > 60)}(T)$ *and* $Q_2 = \sigma_{(A \in (40,80]) \wedge (B \leq 20)}(T)$. *We have* $\mathcal{P}_{QC}(A) = \{[-\infty, 40], (40, 50], (50, 80], (80, \infty]\}$ $\mathcal{P}_{QC}(B) = \{[-\infty, 20], (20, 60], (60, \infty]\}$, *and* $\mathcal{P}_{QC}(C) = \{[-\infty, \infty]\}$. □

Tuple Class: Definition

Given a relation $T(A_1, \cdots, A_n)$ and a set of queries QC, a *tuple class* for T relative to QC is defined as a tuple of subsets $(I_1, \cdots, I_n)$ where each $I_j \in \mathcal{P}_{QC}(A_j)$. We say that a *tuple* $t \in T$ *belongs to a tuple class* $TC = (I_1, \cdots, I_n)$, denoted by $t \in TC$, if $t.A_j \in I_j$ for each $j \in [1, n]$.

EXAMPLE 5.3. *Continuing with Example 5.1,* $TC = ((40, 50], [-\infty, 20], [-\infty, \infty])$ *is an example of a tuple class for* T, *and* $(48, 3, 25) \in TC$. □

- A single tuple modification can be represented by a pair (s, d) of tuple classes where a tuple t in s is modified to a tuple t' in d
  - s and d should not be equal
- Possible modifications by a set of (STC, DTC) pairs
  - STC = Source Tuple Class
  - DTC = Destination Tuple Class

Tuple class: observation

- Given D, a set of queries QC
- If D' is generated by modifying n distinct tuples
- D' can partition QC into at most 4^n equivalent query subsets

- Intuition: for every tuple being changed from t to t' and for each query Q in QC
  - both t, t' match Q
  - neither match Q
  - t matches Q, t' does not
  - t' matches Q, t does not
- Extend the notions of cost/balance/minedit to (STC, DTC) pairs

Heuristic

- Search in a smaller domain of "tuple-class pairs"
- Input: a set of candidate queries QC
- Output: A modified database D' with a small value of cost(D')

- Step 1: Generate a skyline (?) SP of (STC, DTC) pairs (s, d) w.r.t. balance(..) and minEdit(..)
- Step 2: Select A "good" subset $S_{OPT} \subseteq$ SP
- Generate D' from D and S_{OPT}

Summary

- Database usability is as important as capability
  - help user formulate query with examples
  - minimize user interaction and time
- Next two lectures: crowd sourcing
  - "wisdom of crowd" is used to implement database operators