DeeBee

Hawk Weisman

Introduction

Background

Deletional

Query Language

Query Processin

Query Flocess

Design

Implementati

Query Parsin

Owner Deservation

Query Processii

DeeBee

Implementation of a Relational Database Query-Processing System

Hawk Weisman

Department of Computer Science Allegheny College

December 8th, 2014

Outline

DeeBee

Hawk Weisman

Introduction

Background

Databases
Ouery Languages

Query Language: Query Processing

DeeRee

Design Implementation Query Parsin

Query Parsing
Parsing Demo
Query Processing
Query Processing
Demo

- 1 Introduction
- 2 Background
 - Relational Databases
 - Query Languages
 - Query Processing
- 3 DeeBee
 - Design
 - Implementation
 - Query Parsing
 - Parsing Demo
 - Query Processing
 - Query Processing Demo

DeeBee

Hawk Weisman

Introductio

Background

Databases

Query Languages

Query Language Query Processin

DeeBee

Implementation
Query Parsing
Parsing Demo

Query Processing

Query processing:

- An application of many concepts from compilers
- Vital to today's world (databases are everywhere)

■ DeeBee:

- A very small relational database (<1500 LoC)
- Implements a subset of the Structured Query Language
- For educational purposes only (don't use this in production)
- Written in the Scala programming language

DeeBee

Hawk Weisman

Introductio

Background Relational Databases

Query Language Query Processin

DeeBee

Implementation
Query Parsing
Parsing Demo
Query Processing

Query processing:

- An application of many concepts from compilers
- Vital to today's world (databases are everywhere)

■ DeeBee:

- A very small relational database (<1500 LoC)
- Implements a subset of the Structured Query Language
- For educational purposes only (don't use this in production)
- Written in the Scala programming language

DeeBee

Hawk Weisman

Query processing:

- An application of many concepts from compilers
- Vital to today's world (databases are everywhere)

- A very small relational database (<1500 LoC)

- Written in the Scala programming language

DeeBee

Hawk Weisman

Query processing:

- An application of many concepts from compilers
- Vital to today's world (databases are everywhere)

- A very small relational database (<1500 LoC)

- Written in the Scala programming language

DeeBee

Hawk Weisman

Query processing:

- An application of many concepts from compilers
- Vital to today's world (databases are everywhere)

- A very small relational database (<1500 LoC)

- Written in the Scala programming language

DeeBee

Hawk Weisman

Introduction

Relational Databases

Query Language Query Processin

Design Implementation

Query Parsing Parsing Demo Query Processing Query Processing

Query processing:

- An application of many concepts from compilers
- Vital to today's world (databases are everywhere)

■ DeeBee:

- A very small relational database (<1500 LoC)
- Implements a subset of the Structured Query Language
- For educational purposes only (don't use this in production)
- Written in the Scala programming language

DeeBee

Hawk Weisman

Introductio

Relational Databases

Query Language Query Processing

Design Implementation Query Parsing Parsing Demo

Query Parsing
Parsing Demo
Query Processing
Query Processing
Demo

Query processing:

- An application of many concepts from compilers
- Vital to today's world (databases are everywhere)

■ DeeBee:

- A very small relational database (<1500 LoC)
- Implements a subset of the Structured Query Language
- For educational purposes only (don't use this in production)
- Written in the Scala programming language

DeeBee

Hawk Weisman

Introductio

Background Relational Databases Query Languages

Query Language
Query Processin

Design Implementation Query Parsing Parsing Demo Query Processing Query Processing

Query processing:

- An application of many concepts from compilers
- Vital to today's world (databases are everywhere)

- A very small relational database (<1500 LoC)
- Implements a subset of the Structured Query Language
- For educational purposes only (don't use this in production)
- Written in the Scala programming language

Outline

DeeBee

Hawk Weisman

Relational Databases

- Background
 - Relational Databases
 - Query Languages
 - Query Processing
- - - Query Parsing
 - Parsing Demo
 - Query Processing
 - Query Processing Demo

DeeBee

Hawk Weisman

Relational Databases

"The primary data model for commercial data-processing applications." [13, 39]

- A database consists of multiple tables of values, called
- A relation consists of: [13, 8, 4]
 - a set of rows, or tuples
 - a set of columns, or attributes
- So how does this relate to compilers?

DeeBee

Hawk Weisman

Relational

Databases

- "The primary data model for commercial data-processing applications." [13, 39]
- A database consists of multiple tables of values, called relations [13, 8, 4]
- A relation consists of: [13, 8, 4]
 - a set of rows, or tuples
 - a set of columns, or attributes
- So how does this relate to compilers?

DeeBee

Hawk Weisman

Introductio

Background

Databases Query Language

Query Language Query Processin

DooPoo

Design

Implementation
Query Parsing
Parsing Demo
Query Processing
Query Processing

"The primary data model for commercial data-processing applications." [13, 39]

- A database consists of multiple tables of values, called relations [13, 8, 4]
- A relation consists of: [13, 8, 4]
 - a set of rows, or tuples
 - a set of columns, or attributes
- So how does this relate to compilers?

DeeBee

Hawk Weisman

Introductio

Background

Databases

Query Language Query Processin

Query Processi

Design

Implementation
Query Parsing
Parsing Demo
Query Processing
Query Processing
Demo

- "The primary data model for commercial data-processing applications." [13, 39]
- A database consists of multiple tables of values, called relations [13, 8, 4]
- A relation consists of: [13, 8, 4]
 - a set of rows, or tuples
 - a set of columns, or attributes
- So how does this relate to compilers?

DeeBee

Hawk Weisman

Introductio

Background

Databases
Query Language

Query Language Query Processin

Dee Dee

Design
Implementation
Query Parsing
Parsing Demo
Query Processing

 "The primary data model for commercial data-processing applications." [13, 39]

- A database consists of multiple tables of values, called relations [13, 8, 4]
- A relation consists of: [13, 8, 4]
 - a set of rows, or tuples
 - a set of columns, or attributes
- So how does this relate to compilers?

Outline

DeeBee

Hawk Weisman

Query Languages

- Background
 - Relational Databases
 - Query Languages
 - Query Processing
- - - Query Parsing
 - Parsing Demo
 - Query Processing
 - Query Processing Demo

DeeBee

Hawk Weisman

Introductio

Background

Query Languages

Query Processing

Design Implementation Query Parsing Parsing Demo Query Processing Query Processing

■ Users and client software interact with databases through query languages [13, 8, 4]

- These are domain-specific languages for accessing and modifying the database
- Query languages are declarative rather than imperative [13, 8, 4]
- Just like other programming languages, query languages must be parsed, analyzed, and compiled or interpreted. [13, 8, 4]

DeeBee

Hawk Weisman

Query Languages

- Users and client software interact with databases through query languages [13, 8, 4]
- These are domain-specific languages for accessing and modifying the database
- Query languages are declarative rather than
- Just like other programming languages, query

DeeBee

Hawk Weisman

Query Languages

- Users and client software interact with databases through query languages [13, 8, 4]
- These are domain-specific languages for accessing and modifying the database
- Query languages are declarative rather than imperative [13, 8, 4]
- Just like other programming languages, query

DeeBee

Hawk Weisman

Introductio

Background
Relational
Databases
Query Languages

DeeBee
Design
Implementation
Query Parsing
Parsing Demo
Query Processin

- Users and client software interact with databases through query languages [13, 8, 4]
- These are domain-specific languages for accessing and modifying the database
- Query languages are declarative rather than imperative [13, 8, 4]
- Just like other programming languages, query languages must be parsed, analyzed, and compiled or interpreted. [13, 8, 4]

SQL

DeeBee

Hawk Weisman

Introduction

Background Relational

Query Languages

Query Processing

Query i locessi

Design

Implementation Query Parsing

Parsing Demo
Query Processin
Query Processin

SQL is the Structured Query Language.

- It is the query language used by most modern RDBMSs
- SQL consists of two components:
 - Data definition language (DDL): defines the structure of the database [13, 8]
 - creating and deleting tables
 - adding relationships between tables
 - et cetera
 - Data manipulation language (DML): accesses and modifies data stored in the database [13, 8]
 - selecting rows
 - adding, deleting, and modifying rows
 - et cetera
- SQL = DDL + DML

SQL

DeeBee

Hawk Weisman

Introductio

Background
Relational
Databases

Query Languages

Query Processing

DooRoo

Design

Query Parsing

Parsing Demo
Query Processing
Query Processing

■ SQL is the Structured Query Language.

- It is the query language used by most modern RDBMSs
- SQL consists of two components:
 - Data definition language (DDL): defines the structure of the database [13, 8]
 - creating and deleting tables
 - adding relationships between tables
 - et cetera
 - Data manipulation language (DML): accesses and modifies data stored in the database [13, 8]
 - selecting rows
 - adding, deleting, and modifying rows
 - et cetera
- SQL = DDL + DML

SOL

DeeBee

Hawk Weisman

Query Languages

SQL is the Structured Query Language.

- It is the guery language used by most modern RDBMSs
- SQL consists of two components:
 - Data definition language (DDL): defines the structure of
 - creating and deleting tables

 - et cetera
 - Data manipulation language (DML): accesses and
 - selecting rows

 - et cetera
- SQL = DDL + DML

SQL

DeeBee

Hawk Weisman

Introductio

Background Relational Databases

Query Languages

Query Processir

Design

Query Parsing
Parsing Demo
Query Processing
Query Processing

SQL is the Structured Query Language.

- It is the query language used by most modern RDBMSs
- SQL consists of two components:
 - Data definition language (DDL): defines the structure of the database [13, 8]
 - creating and deleting tables
 - adding relationships between tables
 - et cetera
 - Data manipulation language (DML): accesses and modifies data stored in the database [13, 8]
 - selecting rows
 - adding, deleting, and modifying rows
 - et cetera
- SQL = DDL + DML

SQL

DeeBee

Hawk Weisman

Introductio

Background
Relational
Databases
Query Languages

Query Languages
Query Processing

Query Processir

Design Implementatio

Query Parsing
Parsing Demo
Query Processing
Query Processing
Demo

- SQL is the Structured Query Language.
- It is the query language used by most modern RDBMSs
- SQL consists of two components:
 - Data definition language (DDL): defines the structure of the database [13, 8]
 - creating and deleting tables
 - adding relationships between tables
 - et cetera
 - Data manipulation language (DML): accesses and modifies data stored in the database [13, 8]
 - selecting rows
 - adding, deleting, and modifying rows
 - et cetera
- SQL = DDL + DML

SOL

DeeBee

Hawk Weisman

Query Languages

SQL is the Structured Query Language.

- It is the guery language used by most modern RDBMSs
- SQL consists of two components:
 - Data definition language (DDL): defines the structure of the database [13, 8]
 - creating and deleting tables
 - adding relationships between tables
 - et cetera
 - Data manipulation language (DML): accesses and modifies data stored in the database [13, 8]
 - selecting rows
 - adding, deleting, and modifying rows
 - et cetera
- SQL = DDL + DML

SQL

DeeBee

Hawk Weisman

Introductio

Background
Relational
Databases
Query Languages

Query Processing

Design Implementation Query Parsing

Query Parsing
Parsing Demo
Query Processing
Query Processing
Demo

- SQL is the Structured Query Language.
- It is the query language used by most modern RDBMSs
- SQL consists of two components:
 - Data definition language (DDL): defines the structure of the database [13, 8]
 - creating and deleting tables
 - adding relationships between tables
 - et cetera
 - Data manipulation language (DML): accesses and modifies data stored in the database [13, 8]
 - selecting rows
 - adding, deleting, and modifying rows
 - et cetera
- SQL = DDL + DML

DeeBee

Hawk Weisman

Query Languages

Example (SQL CREATE TABLE statement (schema))

```
CREATE TABLE Writers (
    id
                   INTEGER NOT NULL PRIMARY KEY,
    first name
                          VARCHAR (15) NOT NULL,
    middle name
                          VARCHAR (15),
    last name
                          VARCHAR (15) NOT NULL,
    birth_date
                          VARCHAR (10) NOT NULL,
                          VARCHAR (10),
    death date
    country_of_origin
                          VARCHAR (20) NOT NULL
);
```

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processing

Query 1 10003311

Design

Implementation
Query Parsing

Parsing Demo
Query Processing

Example (SQL SELECT statement)

```
SELECT * FROM test;
SELECT test1, test2 FROM test;
SELECT * FROM test WHERE test1 = 9 AND test2 = 5;
SELECT * FROM test LIMIT 5;
```

Example (SQL DELETE statement)

```
DELETE FROM test WHERE test2 > 3 LIMIT 100;
```

Example (SQL INSERT statement)

```
INSERT INTO test VALUES (
  1, 'a string', 2, 'another string'
);
```

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processing

- -

Design Implementation

Query Parsing Parsing Demo Query Processing

Example (SQL SELECT statement)

```
SELECT * FROM test;
SELECT test1, test2 FROM test;
SELECT * FROM test WHERE test1 = 9 AND test2 = 5;
SELECT * FROM test LIMIT 5;
```

Example (SQL DELETE statement)

DELETE FROM test WHERE test2 > 3 LIMIT 100;

Example (SQL INSERT statement)

```
INSERT INTO test VALUES (
  1, 'a string', 2, 'another string');
```

DeeBee

Hawk Weisman

Introduction

Background

Polotional

Query Languages

Query Processing

D - - D - -

Design Implementation Query Parsing

Query Parsing
Parsing Demo
Query Processing
Query Processing
Demo

Example (SQL SELECT statement)

```
SELECT * FROM test;
SELECT test1, test2 FROM test;
SELECT * FROM test WHERE test1 = 9 AND test2 = 5;
SELECT * FROM test LIMIT 5;
```

Example (SQL DELETE statement)

```
DELETE FROM test WHERE test2 > 3 LIMIT 100;
```

Example (SQL INSERT statement)

```
INSERT INTO test VALUES (
  1, 'a string', 2, 'another string'
);
```

Outline

DeeBee

Hawk Weisman

Query Processing

Background

- Relational Databases
- Query Languages
- Query Processing
- - - Query Parsing
 - Parsing Demo
 - Query Processing
 - Query Processing Demo

Steps in Query Evaluation

DeeBee

Hawk Weisman

Introduction

Background

5.1.1.

Query Language

Query Processing

Deebe

Implementa

Query Parsing

Parsing Demo Query Processi

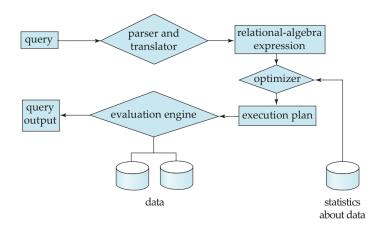


Figure: Steps in query processing [13, 583].

Query Processing

DeeBee

Hawk Weisman

Query Processing

A query processor is essentially a compiler!

- Some stages in the query evaluation process
 - Parsing
 - Semantic analysis

 - Optimization

Query Processing

DeeBee

Hawk Weisman

Introductio

Background

Relational

Query Language

Query Processing

DeeBee

Design

Implementatio

Query Parsing

Query Processing

- A query processor is essentially a compiler!
- Some stages in the query evaluation process correspond directly to those in compilation:
 - Parsing
 - Semantic analysis
 - IR generation (Relational algebra expression)
 - Optimization

DeeBee

Hawk Weisman

Introductio

Background

Databases Output Language

Query Languages

Query Processing

DeeRee

Design

Query Parsing

Query Processing

- A query processor is essentially a compiler!
- Some stages in the query evaluation process correspond directly to those in compilation:
 - Parsing
 - Semantic analysis
 - IR generation (Relational algebra expression)
 - Optimization

DeeBee

Hawk Weisman

Query Processing

- A query processor is essentially a compiler!
- Some stages in the query evaluation process correspond directly to those in compilation:
 - Parsing
 - Semantic analysis

 - Optimization

DeeBee

Hawk Weisman

Introductio

Relational Databases Query Language:

Query Processing

Implementation
Query Parsing
Parsing Demo
Query Processi

A query processor is essentially a compiler!

- Some stages in the query evaluation process correspond directly to those in compilation:
 - Parsing
 - Semantic analysis
 - IR generation (Relational algebra expression)
 - Optimization

DeeBee

Hawk Weisman

Introductio

Relational Databases Query Language

Query Processing

Design Implementation Query Parsing Parsing Demo Query Processi A query processor is essentially a compiler!

- Some stages in the query evaluation process correspond directly to those in compilation:
 - Parsing
 - Semantic analysis
 - IR generation (Relational algebra expression)
 - Optimization

DeeBee

Hawk Weisman

Introductio

Relational Databases Query Language

Query Processing

Design Implementation Query Parsing Parsing Demo Query Processi A query processor is essentially a compiler!

- Some stages in the query evaluation process correspond directly to those in compilation:
 - Parsing
 - Semantic analysis
 - IR generation (Relational algebra expression)
 - Optimization

Outline

DeeBee

Hawk Weisman

Introduction

Background

Relational Databases Query Languages

Query Languages Query Processing

DeeBe

Design

Implementation
Query Parsing
Parsing Demo
Query Processing
Query Processing

1 Introduction

- 2 Background
 - Relational Databases
 - Query Languages
 - Query Processing
- 3 DeeBee
 - Design
 - Implementation
 - Query Parsing
 - Parsing Demo
 - Query Processing
 - Query Processing Demo

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processing

Design

Implem

Query Parsing
Parsing Demo

Query Processin

DeeBee implements a subset of SQL

- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...)
 - LIMIT clauses
 - No JOINS
 - INSERT statements
 - DELETE statements
 - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table
 - Nested predicates (WHERE ... AND ...)

 - No JOINS
 - - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS

DeeBee

Hawk Weisman

Introduction

Background

Daorigioanic

O.....

Query Eanguages

Query Processing

Design

Implemen

Query Parsin

Parsing Demo

Query Processin

DeeBee implements a subset of SQL

- Chosen to balance functionality with time constraints
 - SELECT statements

```
■ Projections (SELECT a, b FROM ...)
```

- Filtering by predicates (SELECT * FROM table WHERE ...)
- Nested predicates (WHERE ... AND ...)
- LIMIT clauses
- No Joins
- INSERT statements
- DELETE statements
 - WHERE and LIMIT clauses
 - Same implementation as SELECT
- CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table
 - Nested predicates (WHERE ... AND ...)

 - No JOINS

 - - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS

DeeBee

Hawk Weisman

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)

 - No JOINS

 - - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...

 - No JOINS

 - - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

Introduction

Background

_ . . .

Query Languages

Query Processing

DeeBee

Design

Ouena Remin

Query Parsing Parsing Demo

Query Processing

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...)
 - LIMIT clauses
 - No JOINS
 - INSERT statements
 - DELETE statements
 - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

Introduction

Background

Relational Databases

Query Languages

Query Processing

Design

Implementation

Query Parsing Parsing Demo Query Processin

Query Processing Query Processing Demo

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...)
 - LIMIT clauses
 - No JOINS
 - INSERT statements
 - DELETE statements
 - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...
 - TITMIT clauses
 - No JOINS
 - INSERT statements
 - - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...
 - TITMIT clauses
 - No JOINS
 - INSERT statements
 - DELETE statements
 - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

Introduction

Background

Relational Databases

Query Languages

Query Processing

Query Processing

Design

Implementation Query Parsing Parsing Demo

Parsing Demo

Query Processing

Query Processing

Demo

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...)
 - LIMIT clauses
 - No JOINS
 - INSERT statements
 - DELETE statements
 - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

Introduction

Background

Databases
Query Languages

Query Processing

Design

Implementa

Query Parsing
Parsing Demo
Query Processing
Query Processing

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...)
 - LIMIT clauses
 - No JOINS
 - INSERT statements
 - DELETE statements
 - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

Introduction

Background

Databases
Query Languages

Query Languages
Query Processing

DooPoo

Design

Implementation
Query Parsing
Parsing Demo
Query Processing
Query Processing

DeeBee implements a subset of SQL

- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...)
 - LIMIT clauses
 - No JOINS
 - INSERT statements
 - DELETE statements
 - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

Introduction

Background

Databases
Query Languages

Query Languages
Query Processing

Design

Implem

Implementation
Query Parsing
Parsing Demo
Query Processing
Query Processing

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...)
 - LIMIT clauses
 - No JOINS
 - INSERT statements
 - DELETE statements
 - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

Introduction

Background

Databases
Query Languages

Query Processing

Design

Implen

Implementation
Query Parsing
Parsing Demo
Query Processing
Query Processing

- DeeBee implements a subset of SQL
- Chosen to balance functionality with time constraints
 - SELECT statements
 - Projections (SELECT a, b FROM ...)
 - Filtering by predicates (SELECT * FROM table WHERE ...)
 - Nested predicates (WHERE ... AND ...)
 - LIMIT clauses
 - No JOINS
 - INSERT statements
 - DELETE statements
 - WHERE and LIMIT clauses
 - Same implementation as SELECT
 - CREATE TABLE and DROP TABLE statements
 - No CHECK constraints
 - No TRIGGERS



DeeBee

Hawk Weisman

Introduction

Background

Relational

Ouery Languages

Query Processing

Design

Design

Ouera Berein

Parsing Demo

■ The actors model [6, 7, 1]

- A construct for concurrent programming
- Actors communicate through message passing
- Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
- Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework

イロト イ団ト イヨト イヨト 手口 りゅぐ

DeeBee

Hawk Weisman

Introduction

Background

Bolotional

Query Language

Query Processing

Query Frocessin

Design

Implemental

Query Parsing

Parsing Demo Query Processing

- A construct for concurrent programming
- Actors communicate through message passing
- Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
- Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework



DeeBee

Hawk Weisman

Introduction

Background

5 1 11 1

Query Languages

Oughy Proceeding

Query Processing

Design

Implementati

Query Parsing

Pareing Domo

Query Processin

- A construct for concurrent programming
- Actors communicate through message passing
- Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
- Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework



DeeBee

Hawk Weisman

Design

- A construct for concurrent programming
- Actors communicate through message passing
- Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
- Actors engueue recieved messages and respond to
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka イロト イ団ト イヨト イヨト 手口 りゅぐ

DeeBee

Hawk Weisman

Design

- A construct for concurrent programming
- Actors communicate through message passing
- Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
- Actors engueue recieved messages and respond to
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka イロト イ団ト イヨト イヨト 手口 りゅぐ

DeeBee

Hawk Weisman

Introduction

Background

5

Query Languages

Query Processing

Query Processin

Design

Implementat

Ouery Parsin

Parsing Demo

Query Processin

- A construct for concurrent programming
- Actors communicate through message passing
- Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
- Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework

DeeBee

Hawk Weisman

Design

- A construct for concurrent programming
- Actors communicate through message passing
- Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
- Actors engueue recieved messages and respond to
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka イロト イ団ト イヨト イヨト 手口 りゅぐ

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processing

DeeBee

Design

Ouera Berein

Query Parsing

Query Processin

- A construct for concurrent programming
- Actors communicate through message passing
- Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
- Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processing

DeeBe

Design

Implementation

Query Parsing Parsing Demo

Query Processing

- The actors model [6, 7, 1]
 - A construct for concurrent programming
 - Actors communicate through message passing
 - Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
 - Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processing

DeeBee

Design

Implementation

Query Parsing Parsing Demo

Query Processin
Query Processin

- A construct for concurrent programming
- Actors communicate through message passing
- Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
- Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processing

DooPoo

Design

Implementation

Query Parsing

Query Processing

- The actors model [6, 7, 1]
 - A construct for concurrent programming
 - Actors communicate through message passing
 - Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
 - Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework

DeeBee

Hawk Weisman

- The actors model [6, 7, 1]
 - A construct for concurrent programming
 - Actors communicate through message passing
 - Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
 - Actors engueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka イロト イ団ト イヨト イヨト 手口 りゅぐ

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processing

Design

Implementat

Query Parsin

Parsing Demo

Query Processing Query Processing Demo

- A construct for concurrent programming
- Actors communicate through message passing
- Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
- Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processir

Design

Implementati

Query Parsin

Parsing Demo

Query Processing

Query Processing

Demo

- The actors model [6, 7, 1]
 - A construct for concurrent programming
 - Actors communicate through message passing
 - Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
 - Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

_ _ _

DeeB

Implementat

Query Parsin Parsing Demo

Query Processing Query Processing Demo

- The actors model [6, 7, 1]
 - A construct for concurrent programming
 - Actors communicate through message passing
 - Messages are:
 - Immutable
 - Asynchronous
 - Anonynous (decoupled)
 - Actors enqueue recieved messages and respond to them in order
- Essentially, an actor is a state machine with a mailbox
- Advantages:
 - Fault tolerance (loose coupling)
 - Scalability
 - Concurrency
 - Event-driven (good for databases)
- In Scala, the Actors model is provided by the Akka framework



DeeBee

Hawk Weisman

Introduction

Background

Dackground

Ouen/Language

Query Processin

D D

Design

Implementa

Query Parsin

Parsing Demo Query Processin

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables
 - sending query results to the querying entity
 - creating and deleting table actors
- A table actor is responsible for:
 - recieving queries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Introduction

Backgroung

Relational

Query Language

Query Processing

Design

Implement

Query Parsing

Parsing Demo

Query Processin

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables
 - sending query results to the querying entity
 - creating and deleting table actors
- A table actor is responsible for:
 - recieving queries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Introduction

Background

5 1 11 1

Ouery Language

Query Processin

DeeBe

Design

Implementation

Query Parsing

Query Processing

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables
 - sending query results to the querying entity
 - creating and deleting table actors
- A table actor is responsible for:
 - recieving queries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Introduction

Rackground

Dackground

Ouen Language

Query Processin

DooPoo

Design

Implementatio

Query Parsin

Parsing Demo

Query Processing

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables
 - sending query results to the querying entity
 - creating and deleting table actors
- A table actor is responsible for:
 - recieving queries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Design

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables

 - creating and deleting table actors
- A table actor is responsible for:
 - recieving gueries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Design

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables
 - sending query results to the querying entity
 - creating and deleting table actors
- A table actor is responsible for:
 - recieving gueries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Design

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables
 - sending query results to the querying entity
 - creating and deleting table actors
- A table actor is responsible for:
 - recieving gueries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Introduction

Background

Databases Query Language

Query Language Query Processin

DeeBe

Design

Implementation
Query Parsing

Parsing Demo
Query Processing

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables
 - sending query results to the querying entity
 - creating and deleting table actors
- A table actor is responsible for:
 - recieving queries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Introduction

Background

Databases Query Language

Query Processin

Design

Implementat

Query Parsing Parsing Demo

Parsing Demo
Query Processing
Query Processing

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables
 - sending query results to the querying entity
 - creating and deleting table actors
- A table actor is responsible for:
 - recieving queries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Design

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables
 - sending query results to the querying entity
 - creating and deleting table actors
- A table actor is responsible for:
 - recieving queries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Design

- tables
- databases
- frontends (connections into the database)
- ...are all represented by actors
- A database actor is responsible for:
 - dispatching queries to its' tables
 - sending query results to the querying entity
 - creating and deleting table actors
- A table actor is responsible for:
 - recieving queries
 - (possibly) updating its' state
 - responding with query results or errors

DeeBee

Hawk Weisman

Introductio

Background

Databases

Query Language:

Query Language Query Processing

Design

Implementation

Query Parsing Parsing Demo Query Processing Query Processing Demo

- SQL queries are internally represented using an abstract syntax tree (AST)
- Connection actors recieve query strings, parse them, and send the AST to the database actor
- Database actor either:
 - processes DDL queries by creating/deleting tables
 - dispatches DML queries to the target child table
- Queries are interpreted (not compiled) against a context

DeeBee

Hawk Weisman

Design

- SQL queries are internally represented using an abstract syntax tree (AST)
- Connection actors recieve query strings, parse them, and send the AST to the database actor.
- Database actor either:
 - processes DDL queries by creating/deleting tables
 - dispatches DML queries to the target child table
- Queries are interpreted (not compiled) against a

DeeBee

Hawk Weisman

Design

- SQL queries are internally represented using an abstract syntax tree (AST)
- Connection actors recieve query strings, parse them, and send the AST to the database actor.
- Database actor either:
 - processes DDL queries by creating/deleting tables
 - dispatches DML queries to the target child table
- Queries are interpreted (not compiled) against a

DeeBee

Hawk Weisman

Introductio

Background
Relational
Databases
Query Languages

Query Language Query Processin

Design

Implementation
Query Parsing
Parsing Demo
Query Processing
Query Processing

- SQL queries are internally represented using an abstract syntax tree (AST)
- Connection actors recieve query strings, parse them, and send the AST to the database actor
- Database actor either:
 - processes DDL queries by creating/deleting tables
 - dispatches DML queries to the target child table
- Queries are interpreted (not compiled) against a context

DeeBee

Hawk Weisman

Design

SQL queries are internally represented using an abstract syntax tree (AST)

- Connection actors recieve query strings, parse them, and send the AST to the database actor.
- Database actor either:
 - processes DDL queries by creating/deleting tables
 - dispatches DML queries to the target child table
- Queries are interpreted (not compiled) against a

DeeBee

Hawk Weisman

Introduction

Relational Databases Query Languages

Query Processi

Design Implementati

Implementation
Query Parsing
Parsing Demo
Query Processing
Query Processing
Demo

SQL queries are internally represented using an abstract syntax tree (AST)

- Connection actors recieve query strings, parse them, and send the AST to the database actor
- Database actor either:
 - processes DDL queries by creating/deleting tables
 - dispatches DML queries to the target child table
- Queries are interpreted (not compiled) against a context

Outline

DeeBee

Hawk Weisman

Implementation

- - Relational Databases
 - Query Languages
 - Query Processing
- DeeBee

 - Implementation
 - Query Parsing
 - Parsing Demo
 - Query Processing
 - Query Processing Demo

DeeBee

Hawk Weisman

Introductio

Background
Relational
Databases
Query Languages

Query Language: Query Processing

Design Implementation

Query Parsing

Parsing Demo Query Processin Query Processin

■ DeeBee's query processor parses queries using combinator parsing [11, 14, 2, 12]

- This is a functional-programming approach to text parsing
 - A parser is a function which accepts some strings and rejects others
 - A parser-combinator is a higher-order function which takes as input two or more parsers and returns combined parser
 - By repeatedly combining simpler parsers into more complex ones, a recursive-descent parser can be created

DeeBee

Hawk Weisman

Introduction

Relational Databases Query Languages

Query Processi

Design Implementation

Query Parsing
Parsing Demo
Query Processin

■ DeeBee's query processor parses queries using combinator parsing [11, 14, 2, 12]

- This is a functional-programming approach to text parsing
 - A parser is a function which accepts some strings and rejects others
 - A parser-combinator is a higher-order function which takes as input two or more parsers and returns combined parser
 - By repeatedly combining simpler parsers into more complex ones, a recursive-descent parser can be created

DeeBee

Hawk Weisman

Introductio

Background
Relational
Databases
Query Languages

Design Implementation

Query Parsing
Parsing Demo

Parsing Demo

Query Processing

Query Processing

Demo

- DeeBee's query processor parses queries using combinator parsing [11, 14, 2, 12]
- This is a functional-programming approach to text parsing
 - A parser is a function which accepts some strings and rejects others
 - A parser-combinator is a higher-order function which takes as input two or more parsers and returns combined parser
 - By repeatedly combining simpler parsers into more complex ones, a recursive-descent parser can be created

DeeBee

Hawk Weisman

Introductio

Background Relational Databases Query Languages

Design

Query Parsing
Parsing Demo
Query Processing
Query Processing

■ DeeBee's query processor parses queries using combinator parsing [11, 14, 2, 12]

- This is a functional-programming approach to text parsing
 - A parser is a function which accepts some strings and rejects others
 - A parser-combinator is a higher-order function which takes as input two or more parsers and returns combined parser
 - By repeatedly combining simpler parsers into more complex ones, a recursive-descent parser can be created

DeeBee

Hawk Weisman

Introductio

Background Relational Databases Query Languages

Design

Query Parsing
Parsing Demo
Query Processing
Query Processing

■ DeeBee's query processor parses queries using combinator parsing [11, 14, 2, 12]

- This is a functional-programming approach to text parsing
 - A parser is a function which accepts some strings and rejects others
 - A parser-combinator is a higher-order function which takes as input two or more parsers and returns combined parser
 - By repeatedly combining simpler parsers into more complex ones, a recursive-descent parser can be created

Parser Combinators in Scala

DeeBee

Hawk Weisman

Introductio

Background

Relational Databases

Query Language Query Processin

Design

Implementation
Query Parsing

Parsing Demo
Query Processin

Scala's parsing library follows the philosophy of embedded DSLs [5, 9, 11, 12]

■ It allows parsers to be specified in BNF-like syntax

Parser Combinators in Scala

DeeBee

Hawk Weisman

Introductio

Background

Databases
Query Language

DeeBee

Design Implementation

Query Parsing
Parsing Demo
Query Processin

 Scala's parsing library follows the philosophy of embedded DSLs [5, 9, 11, 12]

■ It allows parsers to be specified in BNF-like syntax

Parser Combinators in Scala

DeeBee

Hawk Weisman

Introductio

Background
Relational
Databases
Query Language

Design

Implementation

Query Parsing

Parsing Demo Query Processing Query Processing Demo

- Scala's parsing library follows the philosophy of embedded DSLs [5, 9, 11, 12]
- It allows parsers to be specified in BNF-like syntax

DeeBee

Hawk Weisman

Introduction

Background

Dackground

Ouery Language

Query Processin

Query Processir

Design

Implementation

Query Parsing

Parsing Demo

Query Processis

Packrat parsers add a memoization facility [10, 3]

- Guarantees unlimited lookahead and linear parse time
- Allows parsing of left-recursive grammars
- Parser functions are replaced by lazily-evaluated values

```
lazy val expression: P[Expr[_]] =
    ("(" ~> comparison <~ ")") ^^{{
      case c: Comparison => new ParenComparison(c)
}
    | comparison
    | literal
    | identifier
lazy val comparison: P[Comparison] =
    expression ~ operator ~ expression ^^ {
      case lhs ~ op ~ rhs => Comparison(lhs, op, rhs
}
```

DeeBee

Hawk Weisman

Introduction

Background

- . . .

Ouery Language

Query Processin

DooPoo

Design

Query Parsing

Parsing Demo Query Processi Packrat parsers add a memoization facility [10, 3]

- Guarantees unlimited lookahead and linear parse time
- Allows parsing of left-recursive grammars
- Parser functions are replaced by lazily-evaluated values

DeeBee

Hawk Weisman

Introduction

Background

Dackground

Query Language

Query Erocessin

Query Processin

Design

Query Parsing

Parsing Demo

Query Processi Query Processi

- Packrat parsers add a memoization facility [10, 3]
 - Guarantees unlimited lookahead and linear parse time
 - Allows parsing of left-recursive grammars
- Parser functions are replaced by lazily-evaluated values

DeeBee

Hawk Weisman

Introduction

Background

Dackground

Ouen/Language

Query Earlyuage

Query Processi

Design

Query Parsing

Parsing Demo Query Process Packrat parsers add a memoization facility [10, 3]

- Guarantees unlimited lookahead and linear parse time
- Allows parsing of left-recursive grammars
- Parser functions are replaced by lazily-evaluated values

DeeBee

Hawk Weisman

Introduction

Background

Databases
Ouery Language

Query Processin

Design Implementatio

Query Parsing

Query Processin Query Processin Demo

- Packrat parsers add a memoization facility [10, 3]
 - Guarantees unlimited lookahead and linear parse time
 - Allows parsing of left-recursive grammars
- Parser functions are replaced by lazily-evaluated values

Outline

DeeBee

Hawk Weisman

Parsing Demo

- - Relational Databases
 - Query Languages
 - Query Processing
- DeeBee

 - - Query Parsing
 - Parsing Demo
 - Query Processing
 - Query Processing Demo

Outline

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processing

DeeBee

Design Implementation

Query Parsing Parsing Demo

Query Processing

1 Introduction

- 2 Background
 - Relational Databases
 - Query Languages
 - Query Processing
- 3 DeeBee
 - Design
 - Implementation
 - Query Parsing
 - Parsing Demo
 - Query Processing
 - Query Processing Demo

DeeBee

Hawk Weisman

Introduction

Background

Daonground

Databases

Query Language

Query Processing

DeeBe

Design

Query Parsing

Query Processing

Query Processing

DeeBee queries are interpreted

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

Query Languages

Query Processing

Query Processin

Design

Query Parsing

Query Processing

Query Processing

DeeBee queries are interpreted

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

5.1...

Query Languages

Query Processin

Query Processing

Design

Query Parsing Parsing Demo

Query Processing

DeeBee queries are interpreted

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not nul
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

5.1...

Query Languages

Query Processin

Query i rocessii

Design

Implementation Query Parsing

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

Daong. oama

Query Languages

Query Processin

_ _

Design Implementation

Query Parsing Parsing Demo

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

Daorigioanio

Query Languages

Query Processing

Design

Query Parsing
Parsing Demo

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Constraints validation

 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction

DeeBee

Hawk Weisman

Introduction

Background

Daorigioanio

Query Languages

Query Processing

Query Processin

Design

Implementation Query Parsing Parsing Demo

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

Dackground

Query Languages

Query Processing

Query Processin

Design

Query Parsing
Parsing Demo

Query Processing

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure gueries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

Polational

Query Languages

Query Processing

D D

Design Implementation

Query Parsing Parsing Demo

Query Processing

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

Daorigioanio

Query Languages

Query Processing

Query Processin

Design

Implementation

Query Parsing

Parsing Demo

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

5

Query Languages

Query Processing

Query Processin

Design

Implementation
Query Parsing
Parsing Demo

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

5

Query Languages

Query Processin

Query Processin

Design

Implementation
Query Parsing
Parsing Demo

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing

DeeBee

Hawk Weisman

Introduction

Background

Relational

Query Languages

Query Processin

Dee Dee

Implementation

Query Processing

- Interpretation is contextualized against a database
 - Type checking
 - In a compiler, context is preceeding program statements
 - In DBMS, context is the schema of the target table
 - Predicate interpretation
 - Convert AST nodes to Scala partial functions
 - Nested predicates are constructed from leaves to roots
 - Constraints validation
 - Ensure queries don't violate table constraints
 - Uniqueness
 - Not null
 - Type constraints
 - Eventually, this will be deferrable for transaction processing



Outline

DeeBee

Hawk Weisman

Introduction

Background

Relational Databases

Query Languages
Query Processing

DeeBee Design

Implementation
Query Parsing
Parsing Demo
Query Processing
Query Processing
Demo

1 Introduction

2 Background

- Relational Databases
- Query Languages
- Query Processing

3 DeeBee

- Design
- Implementation
 - Query Parsing
- Parsing Demo
- Query Processing
- Query Processing Demo

References I

DeeBee

Hawk Weisman

Appendix References

References

- [1] Gul Abdulnabi Agha. Actors: a model of concurrent computation in distributed systems. 1985.
- [2] Jeroen Fokker. Functional parsers. In *Advanced Functional Programming*, pages 1–23. Springer, 1995.
- [3] Richard A Frost, Rahmatullah Hafiz, and Paul Callaghan. Parser combinators for ambiguous left-recursive grammars. In *Practical Aspects of Declarative Languages*, pages 167–181. Springer, 2008.
- [4] Hector Garcia-Molina, Jeffrey D Ullman, and Jennifer Widom. *Database System Implementation*, volume 654. Prentice Hall Upper Saddle River, NJ:, 2000.
- [5] Debasish Ghosh. *DSLs in action*. Manning Publications Co., 2010.



References II

DeeBee

Hawk Weisman

Appendi:

References

- [6] Munish Gupta. *Akka Essentials*. Packt Publishing Ltd, 2012.
- [7] Philipp Haller. On the integration of the actor model in mainstream technologies: the scala perspective. In Proceedings of the 2nd edition on Programming systems, languages and applications based on actors, agents, and decentralized control abstractions, pages 1–6. ACM, 2012.
- [8] Jan L Harrington. *Relational database design and implementation: clearly explained*. Morgan Kaufmann, 2009.

References III

DeeBee

Hawk Weisman

Appendix References

References

- [9] Christian Hofer, Klaus Ostermann, Tillmann Rendel, and Adriaan Moors. Polymorphic embedding of DSLs. In *Proceedings of the 7th International Conference on Generative Programming and Component Engineering*, pages 137–148. ACM, 2008.
- [10] Manohar Jonnalagedda, Martin Odersky, and Tiark Rompf. Packrat Parsing in Scala. Technical report, Ecole Polytechnique Fédérale de Lausanne, 2009.
- [11] Adriaan Moors, Frank Piessens, and Martin Odersky. Parser combinators in Scala. Technical report, Katholieke Universiteit Leuven, 2008.
- [12] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala*. Artima Inc, 2008.
- [13] A. Silberschatz, H. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill Education, 2010.

References IV

DeeBee

Hawk Weisman

Appendix References

References

[14] S Doaitse Swierstra. Combinator parsers: From toys to tools. *Electronic Notes in Theoretical Computer Science*, 41(1):38–59, 2001.