A Compiler Optimization for Automatic Database Result Caching

Ziv Scully (CMU)
Adam Chlipala (MIT)

POPL’17
Teach For Splash

Teach 2500+ high school students all that you know.

Join hundreds of MIT students in sharing your knowledge.

Past classes include Calculus, Monads to Melody, My Very First Website, Extreme Origami, and more!

Nov 21 and 22, 2015
Register Now at esp.mit.edu/reg
Deadline: October 2
Teach a 1 hr class! (or more)
 Teach For Splash  
Teach 2500+ high school students all that you know.  
Join hundreds of MIT students in sharing your knowledge.  
Past classes include Calculus, Monads to Melody, My Very First Website, Extreme Origami, and more!  
Nov 21 and 22, 2015  
Register Now at esp.mit.edu/reg  
Deadline: October 2  
Teach a 1 hr class! (or more)  

M11038: A Battle of Combinatorics  
Difficulty: **  
Teachers: Luis Herrera Arias  
Come and learn about counting things you didn't know you could count. We'll play fun games and learn the secrets of gambling.  
Meeting Time  
Section 1: Sun 9:05am--11:55am  
Grades  
10 - 12  
Enrollment  
Section 1: Full! (max 30)  

M11106: Counting Beyond Infinity  
Difficulty: ****  
Teachers: Dylan Hendrickson, Jordan Hines  
What if you started counting and never stopped? In this class, we'll talk about ordinals, the numbers you get by doing this. We'll see many types of infinity and do strange and exciting things with them!  
Prerequisites  
Know what it means for a set to be countable/uncountable. Prior experience with proofs and set theory would be helpful.  
Meeting Time  
Section 1: Sun 10:05am--11:55am  
Grades  
9 - 12  
Enrollment  
Section 1: Full! (max 12)  

M11128: Calculate Pi with Trains!  
Difficulty: ***  
Teachers: Ziv Scully  
It turns out that you can calculate pi to very high accuracy by bouncing a small train and a big train into a wall. Come on a journey through Extra-Nice Physics Land (where there's no friction and all collisions are perfectly elastic) to see how it works!  
Prerequisites  
Given the equation of a line, you should know how to find its slope. We'll also use the Pythagorean theorem.  
Meeting Time  
Section 1: Sun 11:05am--11:55am  
Grades  
9 - 12  
Enrollment  
Section 1: Full! (max 40)
<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Credits</th>
<th>Difficulty</th>
<th>Teacher(s)</th>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>M11038</td>
<td>A Battle of Combinatorics</td>
<td>Full</td>
<td>**</td>
<td>Luis Herrera Arias</td>
<td>Know what it means for a set to be countable/uncountable. Prior experience with proofs and set theory would be helpful.</td>
</tr>
<tr>
<td>M11106</td>
<td>Counting Beyond Infinity</td>
<td>Full</td>
<td>****</td>
<td>Dylan Hendrickson, Jordan Hines</td>
<td>What if you started counting and never stopped? In this class, we'll talk about ordinals, the numbers you get by doing this. We'll see many types of infinity and do strange and exciting things with them!</td>
</tr>
<tr>
<td>M11128</td>
<td>Calculate Pi with Trains!</td>
<td></td>
<td>***</td>
<td>Ziv Scully</td>
<td>Given the equation of a line, you should know how to find its slope. We'll also use the Pythagorean theorem.</td>
</tr>
</tbody>
</table>

**Teach For Splash**

Teach 2500+ high school students all that you know.

Join hundreds of MIT students in sharing your knowledge.

Past classes include Calculus, Monads to Melody, My Very First Website, Extreme Origami, and more!

Nov 21 and 22, 2015
Register Now at esp.mit.edu/reg
Deadline: October 2
Teach a 1 hr class! (or more)

**Teach For Splash**

Teach 2500+ high school students all that you know.

Join hundreds of MIT students in sharing your knowledge.

Past classes include Calculus, Monads to Melody, My Very First Website, Extreme Origami, and more!

Nov 21 and 22, 2015
Register Now at esp.mit.edu/reg
Deadline: October 2
Teach a 1 hr class! (or more)
<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
<tr>
<td>id</td>
<td>title</td>
<td>max_size</td>
<td>size</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------</td>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
</tbody>
</table>
show catalog

SELECT id, title WHERE TRUE

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
</tbody>
</table>
Web Server

Request logic

Database

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
</tbody>
</table>
Web Server

Request logic

**SELECT** `max_size, size` **WHERE** `id = 11128`

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
</tbody>
</table>
register 11128

Web Server

Request logic

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
</tbody>
</table>
UPDATE size = size + 1
WHERE id = 11128

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>55</td>
</tr>
</tbody>
</table>
## M11038: A Battle of Combinatorics

<table>
<thead>
<tr>
<th>Difficulty: **</th>
<th>Teachers: Luis Herrera Arias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Come and learn about counting things you didn't know you could count. We'll play fun games and learn the secrets of gambling.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Meeting Time</th>
<th>Grades</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1: Sun 9:05am--11:55am</td>
<td>10 - 12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enrollment</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1: Full! (max 12)</td>
<td></td>
</tr>
</tbody>
</table>

## M11106: Counting Beyond Infinity

<table>
<thead>
<tr>
<th>Difficulty: ****</th>
<th>Teachers: Dylan Hendrickson, Jordan Hines</th>
</tr>
</thead>
<tbody>
<tr>
<td>What if you started counting and never stopped? In this class, we'll talk about ordinals, the numbers you get by doing this. We'll see many types of infinity and do strange and exciting things with them!</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prerequisites</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Know what it means for a set to be countable/uncountable. Prior experience with proofs and set theory would be helpful.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Meeting Time</th>
<th>Grades</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1: Sun 10:05am--11:55am</td>
<td>9 - 12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enrollment</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1: Full! (max 40)</td>
<td></td>
</tr>
</tbody>
</table>

## M11128: Calculate Pi with Trains!

<table>
<thead>
<tr>
<th>Difficulty: ***</th>
<th>Teachers: Ziv Scully</th>
</tr>
</thead>
<tbody>
<tr>
<td>It turns out that you can calculate pi to very high accuracy by bouncing a small train and a big train into a wall. Come on a journey through Extra-Nice Physics Land (where there's no friction and all collisions are perfectly elastic) to see how it works!</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prerequisites</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Given the equation of a line, you should know how to find its slope. We'll also use the Pythagorean theorem.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Meeting Time</th>
<th>Grades</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1: Sun 11:05am--11:55am</td>
<td>9 - 12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enrollment</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1: 54 (max 55)</td>
<td></td>
</tr>
</tbody>
</table>
### M11038: A Battle of Combinatorics

**Difficulty:** **

Teachers: Luis Herrera Arias

Come and learn about counting things you didn't know you could count. We'll play fun games and learn the secrets of gambling.

**Meeting Time**
- Section 1: Sun 9:05am--10:55am

**Grades**
- 10 - 12

**Enrollment**
- Section 1: Full! (max 30)

### M11106: Counting Beyond Infinity

**Difficulty:** ****

Teachers: Dylan Hendrickson, Jordan Hines

What if you started counting and never stopped? In this class, we'll talk about ordinals, the numbers you get by doing this. We'll see many types of infinity and do strange and exciting things with them!

**Prerequisites**
- Know what it means for a set to be countable/uncountable. Prior experience with proofs and set theory would be helpful.

**Meeting Time**
- Section 1: Sun 10:05am--11:55am

**Grades**
- 9 - 12

**Enrollment**
- Section 1: Full! (max 12)

### M11128: Calculate Pi with Trains!

**Difficulty:** ***

Teachers: Ziv Scully

It turns out that you can calculate pi to very high accuracy by bouncing a small train and a big train into a wall. Come on a journey through Extra-Nice Physics Land (where there's no friction and all collisions are perfectly elastic) to see how it works!

**Prerequisites**
- Given the equation of a line, you should know how to find its slope. We'll also use the Pythagorean theorem.

**Meeting Time**
- Section 1: Sun 11:05am--11:55am

**Grades**
- 9 - 12

**Enrollment**
- Section 1: Full! (max 40)
M11038: A Battle of Combinatorics  Full!

Difficulty: **  Teachers: Luis Herrera Arias

Come and learn about counting things you didn’t know you could count. We’ll play fun games and learn the secrets of gambling.

Meeting Time
Section 1: Sun 9:05am--11:55am

Grades 10 - 12
Enrollment
Section 1: Full! (max 12)

M11106: Counting Beyond Infinity  Full!

Difficulty: ****  Teachers: Dylan Hendrickson, Jordan Hines

What if you started counting and never stopped? In this class, we’ll talk about ordinals, the numbers you get by doing this. We’ll see many types of infinity and do strange and exciting things with them!

Meeting Time
Section 1: Sun 10:05am--11:55am

Grades 9 - 12
Enrollment
Section 1: Full! (max 40)

M11128: Calculate Pi with Trains!

Difficulty: ***  Teachers: Ziv Scully

It turns out that you can calculate pi to very high accuracy by bouncing a small train and a big train into a wall. Come on a journey through Extra-Nice Physics Land (where there’s no friction and all collisions are perfectly elastic) to see how it works!

Meeting Time
Section 1: Sat 11:05am--12:55pm, Sat 2:05pm--5:55pm

Prerequisites
Given the equation of a line, you should know how to find its slope. We’ll also use the Pythagorean theorem.

Grades 9 - 12
Enrollment
Section 1: Full! (max 40)

M10775: Axioms of Z

Teach 2500+ high school students all that you know.

Nov 21 and 22, 2015
Register Now at esp.mit.edu/reg
Deadline: October 2
Teach a 1 hr class! (or more)

Teach For Splash

Past classes include Calculus, Monads to Melody, My Very First Website, Extreme Origami, and more!

Join hundreds of MIT students in sharing your knowledge.

Teach For Splash
<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Enrollments</th>
<th>Grades</th>
<th>Difficulty</th>
<th>Teachers</th>
</tr>
</thead>
<tbody>
<tr>
<td>M11038</td>
<td>M11038: A Battle of Combinatorics Full!</td>
<td>Full!</td>
<td>9-12</td>
<td>**</td>
<td>Luis Herrera Arias</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M11106</td>
<td>M11106: Counting Beyond Infinity Full!</td>
<td>Full!</td>
<td>9-12</td>
<td>****</td>
<td>Dylan Hendrickson, Jordan Hines</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M11128</td>
<td>M11128: Calculate Pi with Trains!</td>
<td>Full!</td>
<td>9-12</td>
<td>***</td>
<td>Ziv Scully</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M10775</td>
<td>M10775: Axioms of Z</td>
<td></td>
<td>11-12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description:**

- **M11038: A Battle of Combinatorics Full!**
  - Teachers: Luis Herrera Arias
  - Description: Come and learn about counting things you didn't know you could count. We'll play fun games and learn the secrets of gambling.
  - Meeting Time: Sun 9:05am--11:55am
  - Grades: 10-12
  - Enrollment: Section 1: Full (max 12)

- **M11106: Counting Beyond Infinity Full!**
  - Teachers: Dylan Hendrickson, Jordan Hines
  - Description: What if you started counting and never stopped? In this class, we'll talk about ordinals, the numbers you get by doing this. We'll see many types of infinity and do strange and exciting things with them!
  - Meeting Time: Sun 9:05am--11:55am
  - Grades: 9-12
  - Enrollment: Section 1: Full (max 30)

- **M11128: Calculate Pi with Trains!**
  - Teachers: Ziv Scully
  - Description: It turns out that you can calculate pi to very high accuracy by bouncing a small train and a big train into a wall. Come on a journey through Extra-Nice Physics Land (where there's no friction and all collisions are perfectly elastic) to see how it works!
  - Meeting Time: Sun 9:05am--11:55am
  - Grades: 9-12
  - Enrollment: Section 1: Full (max 40)
M11038: A Battle of Combinatorics  Full!

Teachers: Luis Herrera Arias

Meeting Time
Section 1: Sun 9:05am--10:55am

Grades
10 - 12

Enrollment
Section 1: Full! (max 30)

Prerequisites
It is recommended that students have a background working with electronics (ex. participated in FIRST) but this is not necessary.

Come and learn about counting things you didn't know you could count. We'll play fun games and learn the secrets of gambling.

M11106: Counting Beyond Infinity  Full!

Teachers: Dylan Hendrickson, Jordan Hines

Meeting Time
Section 1: Sun 10:05am--11:55am

Grades
9 - 12

Enrollment
Section 1: Full! (max 40)

Prerequisites
Know what it means for a set to be countable/uncountable. Prior experience with proofs and set theory would be helpful.

Ever wonder what happens if you started counting and never stopped? In this class, we'll talk about ordinals, the numbers you get by doing this. We'll see many types of infinity and do strange and exciting things with them!

M11128: Calculate Pi with Trains!

Teachers: Ziv Scully

Meeting Time
Section 1: Sun 11:05am--11:55am

Grades
9 - 12

Enrollment
Section 1: Full! (max 40)

Prerequisites
Given the equation of a line, you should know how to find its slope. We'll also use the Pythagorean theorem.

It turns out that you can calculate pi to very high accuracy by bouncing a small train and a big train into a wall. Come on a journey through Extra-Nice Physics Land (where there's no friction and all collisions are perfectly elastic) to see how it works!

Teach For Splash

Teach 2500+ high school students all that you know.

Nov 21 and 22, 2015

Register Now at esp.mit.edu/reg

Deadline: October 2

Teach a 1 hr class! (or more)

Join hundreds of MIT students in sharing your knowledge.

Past classes include Calculus, Monads to Melody, My Very First Website, Extreme Origami, and more!
Teach For Splash

Teach 2500+ high school students all that you know.

Join hundreds of MIT students in sharing your knowledge.

Nov 21 and 22, 2015
Register Now at esp.mit.edu/reg
Deadline: October 2

Teach a 1 hr class! (or more)

Past classes include Calculus, Monads to Melody, My Very First Website, Extreme Origami, and more!

Hi: On some of the classes it says "full" and you can't register, but it shows only 1 session is full and other sessions are available; however it still won't let me register for the available sessions. Thanks

M11038: A Battle of Combinatorics Full!
Teachers: Luis Herrera Arias

Prerequisites
We'll learn to make polyhedra (a.k.a. truncated icosahedra for math folks or G60 for chem folks) out of beads and string. By the end of the class, you'll have your own tiny geometric trinket to keep! Depending on time and interest, we might learn to make other geometric things... A fractal dodecahedron? Polyhedral carbon nanotube? The possibilities are (almost) endless.

Meeting Times
Section 1: Sat 3:05pm-4:55pm
Section 2: Sun 10:05am-11:55am

Grades 9 - 12
Enrollment Section 1: Full! (max 30)

M10775: Axioms of Z

Prerequisites
You have a background working with sets, sequences and inductive proofs, but this is not necessary.

Meeting Times
Section 1: Sat 11:05am-12:55pm, Sat 2:05pm-5:55pm

Grades 11 - 12
Enrollment Section 1: 54 (max 55)

M11106: Counting Beyond Infinity
Teachers: Dylan Hendrickson, Jordan Hines

What if you started counting and never stopped? In this class, we'll talk about ordinals, the numbers you get by doing this. We'll see many types of infinity and do strange and exciting things with them!

Prerequisites
Know what it means for a set to be countable/uncountable. Prior experience with proofs and set theory would be helpful.

Meeting Times
Section 1: Sat 11:05am-12:55pm
Section 2: Sat 2:05pm-5:55pm

Grades 9 - 12
Enrollment Section 1: Full! (max 12)

M11128: Calculate Pi with Trains!
Teachers: Ziv Scully

It turns out that you can calculate pi to very high accuracy by bouncing a small train and a big train into a wall. Come on a journey through Extra-Nice Physics Land (where there's no friction and all collisions are perfectly elastic) to see how it works!

Prerequisites
Given the equation of a line, you should know how to find its slope. We'll also use the Pythagorean theorem.

Meeting Time
Section 1: Sun 11:05am-11:55am

Grades 9 - 12
Enrollment Section 1: Full! (max 10)
**M11038: A Battle of Combinatorics**

*Teachers: Luis Herrera Arias*

Come and learn about counting things you didn’t know you could count. We’ll play fun games and learn the secrets of gambling.

**M11106: Counting Beyond Infinity**

*Teachers: Dylan Hendrickson, Jordan Hines*

What if you started counting and never stopped? In this class, we’ll talk about ordinals, the numbers you get by doing this. We’ll see many types of infinity and do strange and exciting things with them!

**Prerequisites**

Know what it means for a set to be countable/uncountable. Prior experience with proofs and set theory would be helpful.

**M11128: Calculate Pi with Trains!**

*Teachers: Ziv Scully*

It turns out that you can calculate pi to very high accuracy by bouncing a small train and a big train into a wall. Come on a journey through Extra-Nice Physics Land (where there’s no friction and all collisions are perfectly elastic) to see how it works!

**Prerequisites**

Given the equation of a line, you should know how to find its slope. We’ll also use the Pythagorean theorem.

**Meeting Time**

Section 1: Sun 11:05am–11:55am

**Grades**

9 - 12

**Enrollment**

Section 1: 54 (max 55)
<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
<tr>
<td>id</td>
<td>title</td>
<td>max_size</td>
<td>size</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------</td>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
<tr>
<td>id</td>
<td>title</td>
<td>max_size</td>
<td>size</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------</td>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
</tbody>
</table>
### Database

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
</tbody>
</table>

### Web Server

<table>
<thead>
<tr>
<th>Size</th>
<th>cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>54</td>
</tr>
</tbody>
</table>

**Request logic**
<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
</tbody>
</table>

The diagram illustrates the flow of requests from the Web Server to the Database, with details on request sizes and cache usage.
<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>54</td>
</tr>
</tbody>
</table>

Database

Web Server

Request logic

Size

cache

11038 ➔ 12
11106 ➔ 40
11128 ➔ 54

register 11128
UPDATE size = size + 1
WHERE id = 11128

```
id    title                        max_size size
11038  “A Battle of Combinatorics” 12       12
11106  “Counting Beyond Infinity”   40       40
11128  “Calculate Pi With Trains!”  55       55
```
Database

WEB SERVER

Request logic

register 11128

update

UPDATE size = size + 1
WHERE id = 11128

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>55</td>
</tr>
</tbody>
</table>

Database
**UPDATE**

```
<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>55</td>
</tr>
</tbody>
</table>
```

**WHERE** $id = 11128$

**Program instrumentation**

```
Web Server
```

- Request logic
  - Size
    - cache
      - 11038 → 12
      - 11106 → 40
      - 11128 → 55

**Database**

- `register 11128`

- `update`

- `UPDATE size = size + 1`
Program instrumentation

**UPDATE**

```sql
UPDATE size 
WHERE id = 11128
```

SQL analysis

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>55</td>
</tr>
</tbody>
</table>
### SQL analysis

**Update**

\[
\text{UPDATE size = size + 1 WHERE id = 11128}
\]

### Cache data structure

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>55</td>
</tr>
</tbody>
</table>

**Register** 11128

**Program instrumentation**

**Web Server**

Request logic

Database

Cache data structure

Program instrumentation

SQL analysis
UPDATE size = size + 1
WHERE id = 11128

DATABASE

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>max_size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>11038</td>
<td>“A Battle of Combinatorics”</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11106</td>
<td>“Counting Beyond Infinity”</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>11128</td>
<td>“Calculate Pi With Trains!”</td>
<td>55</td>
<td>55</td>
</tr>
</tbody>
</table>

WEB SERVER

Request logic

Program instrumentation

Concurrency control

SQL analysis

Cache data structure
## Approaches to Caching

<table>
<thead>
<tr>
<th></th>
<th>Caching</th>
<th>Automatic</th>
<th>Flexible</th>
</tr>
</thead>
<tbody>
<tr>
<td>No caching</td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Manual instrumentation</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
</tr>
<tr>
<td>Library (e.g. ORM)</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
</tr>
</tbody>
</table>
## Approaches to Caching

<table>
<thead>
<tr>
<th>Method</th>
<th>Caching</th>
<th>Automatic</th>
<th>Flexible</th>
</tr>
</thead>
<tbody>
<tr>
<td>No caching</td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Manual instrumentation</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
</tr>
<tr>
<td>Library (e.g. ORM)</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
</tr>
<tr>
<td>Compiler optimization</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Sqlcache

a compiler optimization for caching in the

Ur/Web programming language
Ur/Web compiler

source

type checking

: Sqlcache

inlining

inlining

code generation

executable
Ur/Web compiler

source

<table>
<thead>
<tr>
<th>type checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>inlining</td>
</tr>
</tbody>
</table>

Sqlcache

| inlining |
| code generation |

Program instrumentation

SQL analysis

Cache data structure

Concurrency control

all automatically

for single-server applications
table drawings : {Shape : int, Fill : int}

fun shapesOfFill x =
gallery <- queryX1 (SELECT Shape FROM drawings
  WHERE drawings.Fill = {[x]})
  (fn shape => (* draw it *));
return <xml>Behold: shapes! {gallery}</xml>

fun addDrawing y z =
dml (INSERT INTO drawings (Shape, Fill)
  VALUES ({[y]}, {[z]});
return <xml>Drawing added!</xml>

fun replaceFill y z =
dml (UPDATE drawings SET Fill = {[y]}
  WHERE Fill = {[z]});
return <xml>Fill replaced!</xml>
table drawings : {Shape : int, Fill : int}

fun shapesOfFill x =
gallery <- queryX1 (SELECT Shape FROM drawings
    WHERE drawings.Fill = {[x]})
    (fn shape => (* draw it *));
return <xml>Behold: shapes! {gallery}</xml>

fun addDrawing y z =
dml (INSERT INTO drawings (Shape, Fill)
    VALUES ({{y}}, {{z}}));
return <xml>Drawing added!</xml>

fun replaceFill y z =
dml (UPDATE drawings SET Fill = {[y]}
    WHERE Fill = {[z]});
return <xml>Fill replaced!</xml>
Ur/Web Example

table drawings : {Shape : int, Fill : int}

fun shapesOfFill x =
gallery <- queryX1 (SELECT Shape FROM drawings WHERE drawings.Fill = {[x]})
(fn shape => (* draw it *))
return <xml>Behold: shapes! {gallery}</xml>

fun addDrawing y z =
dml (INSERT INTO drawings (Shape, Fill) VALUES ({{y}}, {{z}}))
return <xml>Drawing added!</xml>

fun replaceFill y z =
dml (UPDATE drawings SET Fill = {[y]} WHERE Fill = {[z]}))
return <xml>Fill replaced!</xml>
Ur/Web Example

table drawings : {Shape : int, Fill : int}

fun shapesOfFill x =
gallery <- queryX1 (SELECT Shape FROM drawings
WHERE drawings.Fill = {[x]})
(fn shape => (* draw it *)));
return <xml>Behold: shapes! {gallery}</xml>

fun addDrawing y z =
dml (INSERT INTO drawings (Shape, Fill)
VALUES ({{y}}, {{z}}));
return <xml>Drawing added!</xml>

fun replaceFill y z =
dml (UPDATE drawings SET Fill = {[y]}
WHERE Fill = {[z]});
return <xml>Fill replaced!</xml>
Ur/Web Example

```javascript

```table``` drawings : {Shape : int, Fill : int}

```fun``` shapesOfFill x =
gallery <- queryX1 (```SELECT``` Shape FROM drawings ```WHERE``` drawings.Fill = {x})
(fn shape => (* draw it *))
return ```<xml>```Behold: shapes! {gallery}</xml>```;

```fun``` addDrawing y z =
dml (```INSERT INTO``` drawings (Shape, Fill)
```VALUES``` ({y}, {z}));
return ```<xml>```Drawing added!</xml>

```fun``` replaceFill y z =
dml (```UPDATE``` drawings ```SET``` Fill = {y}
```WHERE``` Fill = {z});
return ```<xml>```Fill replaced!</xml>
```
Ur/Web Example

table drawings : {Shape : int, Fill : int}

fun shapesOfFill x =
gallery <- queryX1 (SELECT Shape FROM drawings
WHERE drawings.Fill = {{x}})
(fn shape => (* draw it *));
return <xml>Behold: shapes! {gallery}</xml>

fun addDrawing y z =
dml (INSERT INTO drawings (Shape, Fill)
VALUES ({[y]}, {[z]}));
return <xml>Drawing added!</xml>

fun replaceFill y z =
dml (UPDATE drawings SET Fill = {[y]}
WHERE Fill = {[z]});
return <xml>Fill replaced!</xml>
Invalidation for INSERT

\[
\text{SELECT } \text{shape} \text{ WHERE } \text{fill} = x
\]

\[
\text{INSERT } (\text{shape}, \text{fill}) = (y, z)
\]
Invalidation for INSERT

SELECT shape WHERE fill = x

x = ⚫

INSERT (shape, fill) = (y, z)
Invalidation for INSERT

\[
\text{SELECT } \text{shape} \text{ WHERE } \text{fill} = x
\]

\[
x = \begin{array}{c}
\text{Cache} \\
\text{Insertion: (shape, fill)} = (y, z)
\end{array}
\]
Invalidation for INSERT

SELECT `shape` WHERE `fill` = x

INSERT (shape, fill) = (y, z)
Invalidation for INSERT

SELECT shape WHERE fill = x

Cache

\[ x = \text{[shapes]} \rightarrow \text{[shapes]} \]

\[ x = \text{[shapes]} \rightarrow \text{[shapes]} \]

INSERT \((shape, fill) = (y, z)\)
Invalidation for INSERT

**Cache**

```
x = [spiral] →  □  □
```

```
x = [spiral] →  □  □
```

**Database**

```
\text{INSERT } (\text{shape, fill}) = (y, z)
```

```
(y, z) = [	ext{shape}]
```

**SELECT** \text{shape WHERE fill = } x
Invalidation for INSERT

SELECT shape WHERE fill = x

Insertion for \( (\text{shape}, \text{fill}) = (y, z) \)

\( (y, z) = \)
Invalidation for INSERT

SELECT shape WHERE fill = x

Cache

x = [shape]

x = [fill]

Database

INSERT (shape, fill) = (y, z)

(y, z) = [shape]
Invalidation for INSERT

SELECT shape WHERE fill = x

Cache

\[ x = \text{spiral} \quad \rightarrow \quad \square \]
\[ x = \text{circle} \quad \rightarrow \quad \bigcirc \quad \bigcirc \]

Database

INSERT (shape, fill) = (y, z)

\[ (y, z) = \text{pentagon} \]
Invalidation for INSERT

SELECT shape WHERE fill = x

Cache

\[
\begin{align*}
x &= \text{circle} & \rightarrow & \text{pentagon} & \square \\
x &= \text{rectangle} & \rightarrow & \text{circle} & \times
\end{align*}
\]

Database

\[
\text{INSERT } (\text{shape, fill}) = (y, z)
\]

\[
(y, z) = \text{pentagon}
\]
Invalidation for INSERT

SELECT shape WHERE fill = x

Invalidation formula:
\[ \exists (\text{shape}, \text{fill}). \quad \text{fill} = x \lor \text{shape} = y \lor \text{fill} = z \]
Invalidation for INSERT

**SELECT** *shape* **WHERE** *fill* = *x*

<table>
<thead>
<tr>
<th>Cache</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Shape" /></td>
<td><img src="image2" alt="Database" /></td>
</tr>
</tbody>
</table>

**INSERT** *(shape, fill) = (y, z)*

**Invalidation formula:**

\[ \exists (shape, fill). \quad fill = x \land shape = y \land fill = z \]
Invalidation for INSERT

SELECT shape WHERE fill = x

Invalidation formula:

$$\exists (\text{shape}, \text{fill}). \quad \text{fill} = x \land \text{shape} = y \land \text{fill} = z$$
Invalidation for INSERT

SELECT shape WHERE fill = x

INSERT (shape, fill) = (y, z)

Invalidation formula:

\[ \exists (\text{shape}, \text{fill}). \quad \text{fill} = x \land \text{shape} = y \land \text{fill} = z \]
Invalidation for INSERT

SELECT shape WHERE fill = x

INSERT (shape, fill) = (y, z)

Invalidation formula:
\[ \exists (\text{shape}, \text{fill}). \quad \text{fill} = x \land \text{shape} = y \land \text{fill} = z \]
\[ \Rightarrow x = z \]
Invalidation for INSERT

SELECT shape WHERE fill = \( x \)

Invalidation formula:
\[ \exists (shape, fill). \quad fill = x \land shape = y \land fill = z \]

\[ \Rightarrow x = z \]

cache key
Invalidation for INSERT

SELECT shape WHERE fill = \(x\)

\[\begin{align*}
x &= \text{shape} & \rightarrow & \text{fill} &= x \\
&
\end{align*}\]

\(\text{Cache}\)

\(\text{Database}\)

Invalidation formula:

\[\exists (\text{shape, fill}). \quad \text{fill} = x \land \text{shape} = y \land \text{fill} = z\]

\(\Rightarrow \quad x = z\)

\(\text{Cache key}\)

\(\text{known during update}\)
Invalidation for INSERT

SELECT \textit{shape} WHERE \textit{fill} = x

\begin{align*}
x &= \begin{array}{c}
\text{circle} \\
\text{rectangle}
\end{array} & \rightarrow & \begin{array}{c}
\text{pentagon} \\
\text{square}
\end{array} \\
x &= \begin{array}{c}
\text{circle} \\
\text{rectangle}
\end{array} \quad \text{(crossed out)} & \rightarrow & \begin{array}{c}
\text{pentagon} \\
\text{square}
\end{array}
\end{align*}

\text{Invalidation formula:}

\exists (\textit{shape}, \textit{fill}) \Rightarrow \text{fill} = x \land \textit{shape} = y \land \textit{fill} = z

\implies x = z

\text{ inval}(z);
Invalidation for UPDATE

SELECT shape WHERE fill = x

UPDATE fill = y WHERE fill = z
Invalidation for UPDATE

SELECT shape WHERE fill = x

UPDATE fill = y WHERE fill = z
Invalidation for UPDATE

SELECT shape WHERE fill = x

x = \[
\begin{array}{c}
\text{Cache}
\end{array}
\]

\[
\begin{array}{c}
\text{UPDATE fill = y WHERE fill = z}
\end{array}
\]

Database
Invalidation for UPDATE

SELECT shape WHERE fill = x

UPDATE fill = y WHERE fill = z
Invalidation for UPDATE

SELECT shape WHERE fill = x

Cache

x = [shapes]

UPDATE fill = y WHERE fill = z

Database

[Database symbols]
Invalidation for UPDATE

SELECT shape WHERE fill = x

Update fill = y WHERE fill = z

y = ◦  z = ⌂
Invalidation for UPDATE

SELECT shape WHERE fill = x

Cache

UPDATE fill = y  WHERE fill = z

Database
Invalidation for UPDATE

SELECT shape WHERE fill = x

UPDATE fill = y WHERE fill = z

y = ●  z = ⚫
Invalidation for UPDATE

SELECT shape WHERE fill = x

Cache

x = ●

x = ○

UPDATE fill = y WHERE fill = z

Database

y = ○
z = ●
Invalidation for UPDATE

SELECT shape WHERE fill = x

x = \[\text{shape} \]

UPDATE fill = y WHERE fill = z

y = \[\text{shape} \]

z = \[\text{shape} \]
Invalidation for UPDATE

\[
\text{SELECT } \text{shape} \text{ WHERE } \text{fill} = x
\]

Cache

\[
\begin{align*}
x &= \quad \text{\textcolor{red}{\times}} & \quad \text{\textcolor{red}{\times}} \\
\text{\textcolor{red}{\times}} &= \quad \text{\textcolor{red}{\times}} & \quad \text{\textcolor{red}{\times}} \\
\end{align*}
\]

Database

\[
\text{UPDATE } \text{fill} = y \text{ WHERE } \text{fill} = z
\]

\[
\begin{align*}
y &= \quad \text{\textcolor{red}{\times}} & \quad \text{\textcolor{red}{\times}} \\
\text{\textcolor{red}{\times}} &= \quad \text{\textcolor{red}{\times}} & \quad \text{\textcolor{red}{\times}} \\
\end{align*}
\]
Invalidation for UPDATE

SELECT shape WHERE fill = x

Cache
x = □ □
x = ■ ■

UPDATE fill = y WHERE fill = z

y = ◇ ●
z = ■ ■

Database
Invalidation for UPDATE

SELECT shape WHERE fill = x

Cache

$\begin{align*}
x &= \circred \rightarrow \square \\
x &= \bullet \rightarrow \bigcirc
\end{align*}$

Database

UPDATE fill = y WHERE fill = z

$\begin{align*}
y &= \circ \quad z &= \bullet
\end{align*}$

Invalidation formula:

$\exists (\text{shape}, \text{fill}), (\text{shape}', \text{fill}').$

$(\text{fill} = x \lor \text{fill}' = x) \land (\text{fill}' = y \land \text{fill} = z)$
Invalidation for UPDATE

SELECT shape WHERE fill = x

Cache

\[
\begin{align*}
x &= \circ & \hfill & \text{x is not invalidated}
\end{align*}
\]

\[
\begin{align*}
x &= \# & \hfill & \text{x is invalidated}
\end{align*}
\]

UPDATE fill = y WHERE fill = z

\[
\begin{align*}
y &= \circ & \hfill & \text{y is not invalidated}
\end{align*}
\]

\[
\begin{align*}
z &= \# & \hfill & \text{z is invalidated}
\end{align*}
\]

Invalidation formula:

\[
\exists (\text{shape, fill}), (\text{shape}', \text{fill}').
\]

\[
(fill = x \lor fill' = x)
\]

\[
\land (fill' = y \land fill = z)
\]

\[
\Rightarrow x = y \lor x = z
\]
Invalidation for UPDATE

SELECT shape WHERE fill = x

Cache

∪

UPDATE fill = y WHERE fill = z

y = ⊙ z = 

Invalidation formula:

∃ (shape, fill), (shape', fill').

(fill = x ∨ fill' = x)

∧ (fill' = y ∧ fill = z)

⇒ x = y ∨ x = z
Invalidation for UPDATE

SELECT shape WHERE fill = \( x \)

Cache

UPDATE fill = \( y \) WHERE fill = \( z \)

Invalidation formula:
\[ \exists (\text{shape}, \text{fill}), (\text{shape}', \text{fill}'). \]
\[ (\text{fill} = x \lor \text{fill}' = x) \]
\[ \land (\text{fill}' = y \land \text{fill} = z) \]
\[ \Rightarrow x = y \lor x = z \]
Invalidation for UPDATE

SELECT shape WHERE fill = \( x \)

UPDATE fill = \( y \) WHERE fill = \( z \)

Invalidation formula:

\[ \exists (shape, fill), (shape', fill'). \]
\[ (fill = x \lor fill' = x) \land (fill' = y \land fill = z) \]
\[ \Rightarrow \ x = y \lor x = z \]

inval(y); inval(z);
Invalidation for UPDATE

SELECT shape WHERE fill = \( x \)

Cache
\[
\begin{align*}
x &= \circ \\
x &= \square
\end{align*}
\]

\( \text{UPDATE } fill = \_y \text{ WHERE } fill = \_z \)

Database

Invalidation formula:
\[
\exists (\text{shape}, \text{fill}), (\text{shape}', \text{fill}'). \\
(fill = x \land fill' \neq x) \lor (fill \neq x \land fill' = x) \\
\lor (fill = x \land fill' = x \land \text{shape} \neq \text{shape}')
\]
\[
\Rightarrow x = y \lor x = z
\]

inval(y); inval(z);
Compound Cache Keys

SELECT COUNT(*) WHERE fill = x ∧ shape = w

Cache

\[[x, w] = \begin{array}{c}
\text{Pattern 1} \\
\text{Pattern 2}
\end{array}\rightarrow 24

\[[x, w] = \begin{array}{c}
\text{Pattern 3} \\
\text{Pattern 4}
\end{array}\rightarrow 29
Compound Cache Keys

SELECT COUNT(*) WHERE fill = x \land shape = w

\[
\begin{align*}
[x, w] &= \begin{array}{c}
\text{\includegraphics[scale=0.2]{example1.png}}
\end{array} & \rightarrow & 24 \\
[x, w] &= \begin{array}{c}
\text{\includegraphics[scale=0.2]{example2.png}}
\end{array} & \rightarrow & 29
\end{align*}
\]

INSERT \((shape, fill) = (y, z)\) \quad UPDATE \text{\textit{fill} = y} \quad WHERE \text{\textit{fill} = z}
Compound Cache Keys

SELECT COUNT(*) WHERE fill = x ∧ shape = w

Cache

[x, w] = 👏 → 24
[x, w] = 📋 → 29

INSERT (shape, fill) = (y, z)  UPDATE fill = y WHERE fill = z

⇒ x = z ∧ w = y
⇒ inval([z, y]);
Compound Cache Keys

SELECT COUNT(*) WHERE fill = x ∧ shape = w

\[
\begin{align*}
[x, w] &= \mathcal{C} & \rightarrow & 24 \\
[x, w] &= \mathcal{H} & \rightarrow & 29
\end{align*}
\]

INSERT \((shape, fill) = (y, z)\) \quad UPDATE \(fill = y\) WHERE \(fill = z\)

\[
\begin{align*}
x = z \land w &= y \\
\text{inval}([z, y]); & & x = y \lor x = z \\
\text{inval}([y, *]); \text{ inval}([z, *]);
\end{align*}
\]
Cache Data Structure

SELECT COUNT(*) WHERE fill = x \land shape = w

UPDATE fill = y WHERE fill = z
inval([y, *]); inval([z, *]);
Cache Data Structure

SELECT COUNT(*) WHERE fill = x AND shape = w

UPDATE fill = y WHERE fill = z
inval([y, *]); inval([z, *]);
Cache Data Structure

SELECT COUNT(*) WHERE fill = x \wedge shape = w

UPDATE fill = y WHERE fill = z
inval([y, *]); inval([z, *]);

y = \bigcirc  z = \#  

\[ 53 \rightarrow \bigcirc \]
\[ 24 \rightarrow \pentagon \]
\[ 29 \rightarrow \# \]
Cache Data Structure

SELECT COUNT(*) WHERE fill = x \land shape = w

UPDATE fill = y  WHERE fill = z
inval([y, *]); inval([z, *]);

y = ☯  z = ☯
Cache Data Structure

SELECT COUNT(*) WHERE fill = x ^ shape = w

UPDATE fill = y WHERE fill = z
inval([y, *]); inval([z, *]);

y = ● z = ★
Cache Data Structure

SELECT COUNT(*) WHERE fill = x ∧ shape = w

UPDATE fill = y WHERE fill = z
inval([y, *]); inval([z, *]);

4:00  y =  ⨿  z =  ⚡

1:00  1  53  2:00  2  24  3:00  3  29
SELECT COUNT(*) WHERE fill = x ∧ shape = w

UPDATE fill = y WHERE fill = z
inval([y, *]); inval([z, *]);
Cache Data Structure

SELECT COUNT(*) WHERE fill = \( x \) \( \land \) shape = \( w \)

UPDATE fill = \( y \) WHERE fill = \( z \)
inval([\( y \), *]); inval([\( z \), *]);

\[ y = \]regoray \( z = \)sector
Cache Data Structure

SELECT COUNT(*) WHERE fill = x \land \text{shape} = w

UPDATE fill = y \text{ WHERE fill} = z
inval([y, *]); inval([z, *]);

4:00 \quad y = \quad z =

5:00 \quad (w, x) =

1:00 \quad 53

2:00 \quad 24

3:00 \quad 29
Cache Data Structure

SELECT COUNT(*) WHERE fill = x ∧ shape = w

UPDATE fill = y WHERE fill = z
inval([y, *]); inval([z, *]);

4:00  y =  ⋄  z =  ⋄
5:00  (w, x) =  ⬤
Cache Data Structure

SELECT COUNT(*) WHERE fill = x ∧ shape = w

UPDATE fill = y WHERE fill = z
inval([y, *]); inval([z, *]);

4:00 y =  z =
5:00 (w, x) =

1:00  53
5:00  12
3:00  29
Program instrumentation

SQL analysis

Cache data structure

Concurrency control
Program instrumentation

SQL analysis

talked

Cache data structure

talked

Concurrency control
Program instrumentation

SQL analysis

Cache data structure

Concurrency control

Consolidate cached expressions, but avoid introducing new keys.
Consolidate cached expressions, but avoid introducing new keys.

Two global locks per cache: “data” lock and “transaction” lock.
Consolidate cached expressions, but avoid introducing new keys.

Two global locks per cache: “data” lock and “transaction” lock.

Deactivate caches with low hit rate to reduce serialization.
Performance Evaluation

Course application, concurrency test (queries only)

Throughput (requests/second)

- 1-thread server
- 4-thread server
- 8-thread server
- 12-thread server

Ur/Web
Ur/Web with Sqlcache
Performance Evaluation

Course application, writes test (4-thread server)

<table>
<thead>
<tr>
<th>Throughput (requests/second)</th>
<th>0 writes/second</th>
<th>1 write/second</th>
<th>10 writes/second</th>
<th>100 writes/second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ur/Web</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ur/Web with Sqlcache</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sqlcache

caching as a compiler optimization

https://github.com/urweb/urweb
Good question!
Sqlcache vs. Dyncache

Course application, concurrency test (queries only)

Throughput (requests/second)

0 1000

1-thread server 4-thread server 8-thread server 12-thread server

Ur/Web Ur/Web with Dyncache Ur/Web with Sqlcache
Sqlcache vs. Dyncache

Course application, writes test (4-thread server)

Throughput (requests/second)

- 0 writes/second
- 1 write/second
- 10 writes/second
- 100 writes/second

Ur/Web
Ur/Web with Dyncache
Ur/Web with Sqlcache
Supported SQL

- logic, equalities
- all flavors of JOIN
- nested queries: FROM

- arithmetic, inequalities
- COUNT, SUM
- LIMIT, ORDER BY, GROUP BY
- CURRENT_TIMESTAMP

- nested queries: SELECT, WHERE
- cascading triggers
Related Work

Updating materialized views
Blakely et al. (1986)

TxCache
Ports et al. (2010)

Sync Kit
Benson et al. (2010)
Why Ur/Web?

table drawings : {Shape : int, Fill : int}

fun shapesOfFill x =
    gallery <- queryX1 (SELECT Shape FROM drawings
                        WHERE drawings.Fill = [[x]])
    (fn shape => (* draw it *))
    return <xml>Behold: shapes! {gallery}</xml>

fun addDrawing y z =
    dml (INSERT INTO drawings (Shape, Fill)
         VALUES ([[y]], [[z]]))
    return <xml>Drawing added!</xml>

fun replaceFill y z =
    dml (UPDATE drawings SET Fill = [[y]]
         WHERE Fill = [[z]])
    return <xml>Fill replaced!</xml>
Why Ur/Web?

```plaintext
table drawings : {Shape : int, Fill : int}

fun shapesOfFill x =
gallery <- queryX1 (SELECT Shape FROM drawings WHERE drawings.Fill = {[x]})
(fn shape => (* draw it *))
return <xml>Behold: shapes! {gallery}</xml>

fun addDrawing y z =
dml (INSERT INTO drawings (Shape, Fill) VALUES ({[y]}, {[z]}));
return <xml>Drawing added!</xml>

fun replaceFill y z =
dml (UPDATE drawings SET Fill = {[y]} WHERE Fill = {[z]}));
return <xml>Fill replaced!</xml>
```

First-class SQL
Why Ur/Web?

Controlled side effects

First-class SQL

```plaintext
fun addDrawing y z =
  dml (INSERT INTO drawings (Shape, Fill)
       VALUES ({[y]}, {[z]}));
  return <xml>Drawing added!</xml>

fun replaceFill y z =
  dml (UPDATE drawings SET Fill = {[y]}
       WHERE Fill = {[z]});
  return <xml>Fill replaced!</xml>
```
Why Ur/Web?

Controlled side effects

```
gallery <- queryX1 (SELECT Shape FROM drawings WHERE drawings.Fill = {[x]})
    (fn shape => (* draw it *));
return <xml>Behold: shapes! {gallery}</xml>
```

Lots of inlining

```
fun addDrawing y z =
    dml (INSERT INTO drawings (Shape, Fill)
         VALUES ({[y]}, {[z]}));
return <xml>Drawing added!</xml>
```

First-class SQL

```
fun replaceFill y z =
    dml (UPDATE drawings SET Fill = {[y]}
         WHERE Fill = {[z]}));
return <xml>Fill replaced!</xml>
```