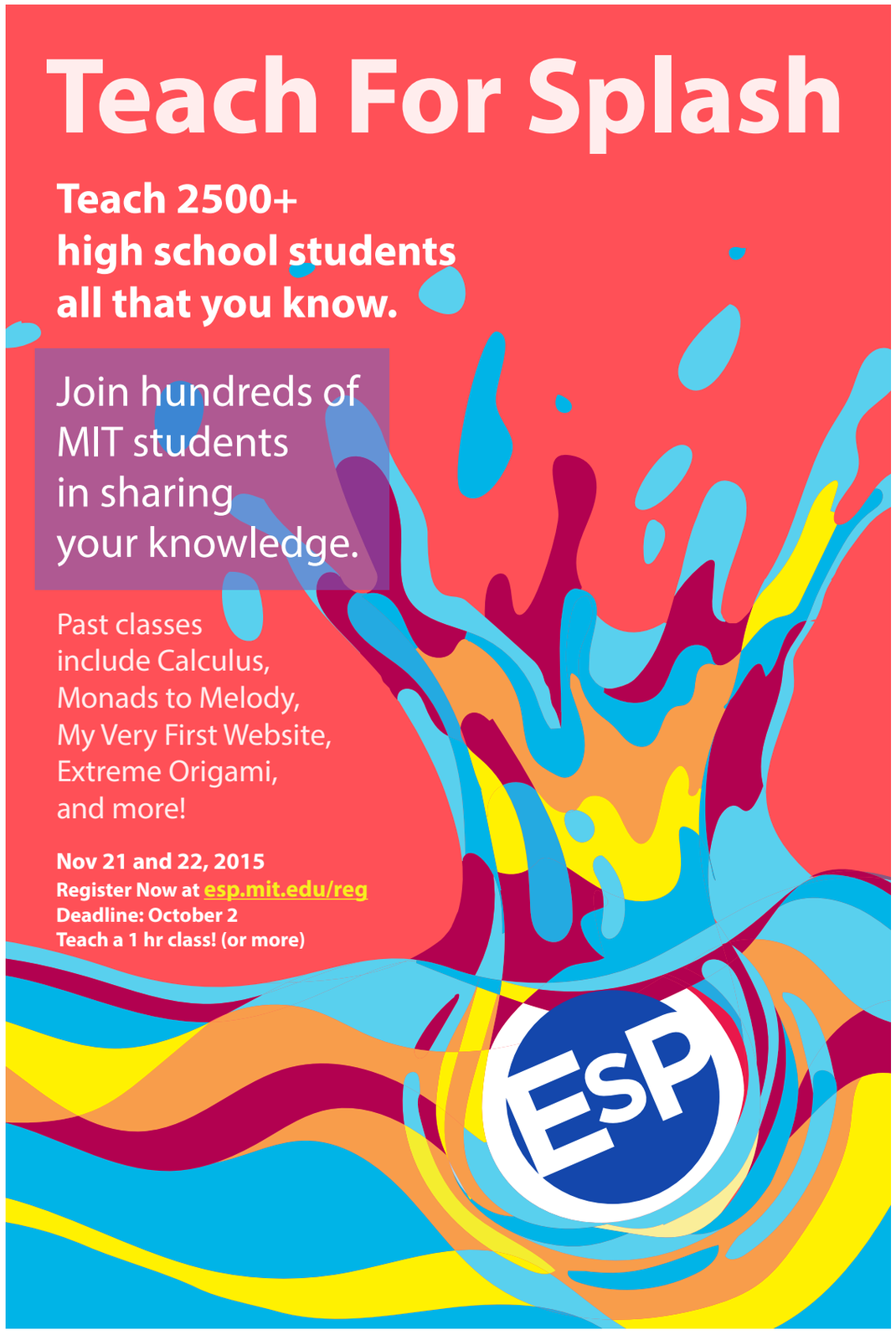
# A Compiler Optimization for Automatic Database Result Caching

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)
ESP


## 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!

**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 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!

**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)


## 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!

**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 40)

## M11128: Calculate Pi with Trains!

Difficulty: ***

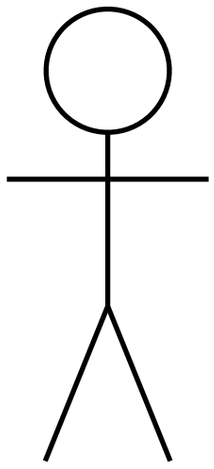
Teachers: Ziv Scully

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!

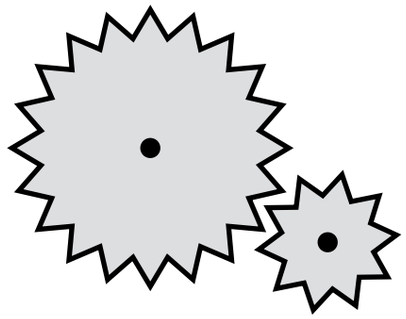
**Prerequisites**
*Given the equation of a line, you should know how to find its slope. We'll also use the Pythagorean theorem.*

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

**Grades**
9 - 12
**Enrollment**
Section 1: 54 (max 55)



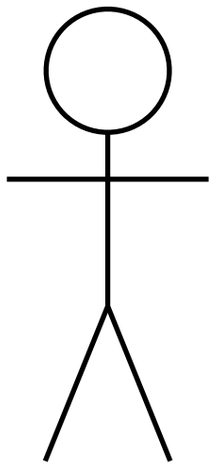
Web Server



Request logic

Database

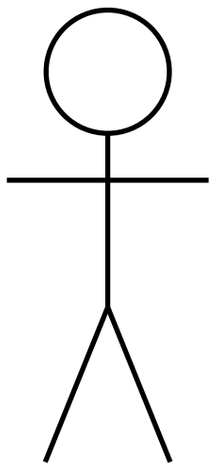
| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |



show catalog

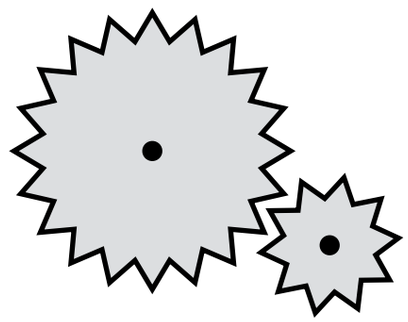

Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |



show catalog

Web Server



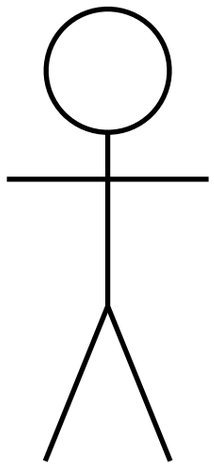
Request logic

query

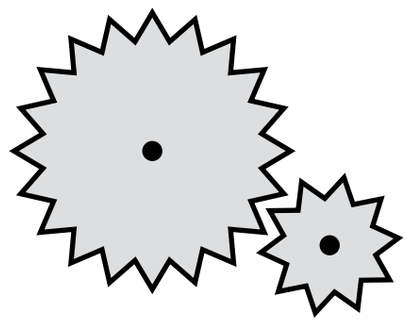
SELECT *id*, *title* WHERE TRUE

Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |



check 11128

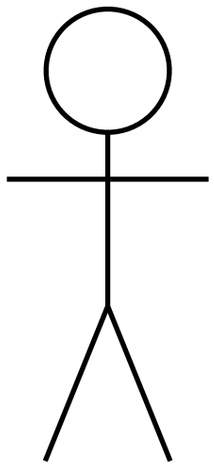


Web Server

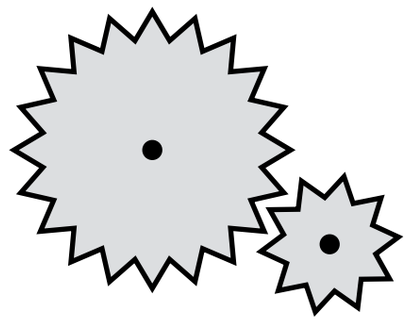
Request logic

Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |



check 11128



Web Server

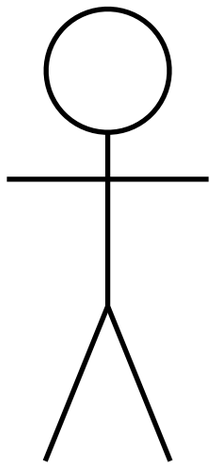
Request logic

query

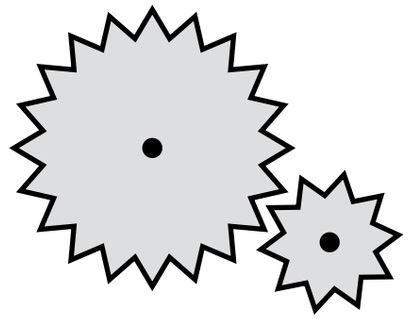
SELECT *max_size*, *size*
WHERE *id* = 11128

Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |



register 11128

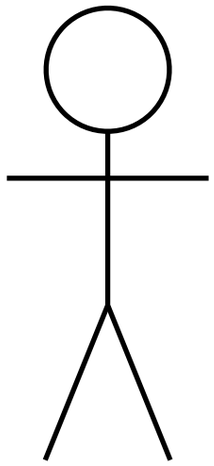


Web Server

Request logic

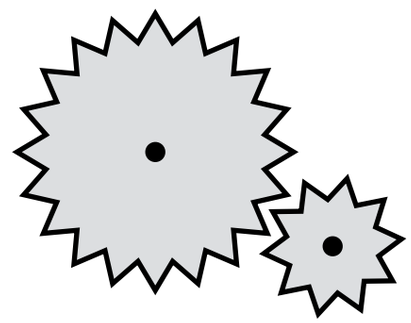
Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |



register 11128

Web Server



Request logic

update

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

Database

| *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** |


## 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!

**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 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!

**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)

# Teach For Splash

**Teach 2500+ high school students all that you know**

Jo
M
in
you

Past c
includ
Monac
My Very
Extreme
and mor

**Nov 21 and**
**Register Now**
**Deadline: Octo**
**Teach a 1 hr cla**

## S9802: How to make a Tesla Coil

Difficulty: ***

Teachers: peter Krogen

Ever wonder how a Tesla coil works? Interested in how you can make 10'+ sparks, and how those sparks can play music? Want to build your very own Tesla coil? Then this class is for you! We'll begin with a quick lecture on how a Tesla Coil works, from the bottom up. We'll then provide you with kits to build solid-state Tesla Coils that can make music, and we'll be there to assist you in building and debugging. The end product is a shoe box sized Tesla coil, capable of generating sparks and playing music.

**Prerequisites**
*It is recommended that students have a background working with electronics (ex, participated in FIRST) 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: 15 (max 16)

*Class temporarily full; please check back later*

## M11038: A Battle of Combinatorics Full!

Difficulty: **

Teachers: Luis Herrera Arias

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

**Meeting Time**
…un 9:05am--11:55am

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

## …nting Beyond Infinity Full!

…lty: ****

Teachers: Dylan Hendrickson, Jordan Hines

…ting and never stopped? In this class, we'll talk about ordinals, the …g this. We'll see many types of infinity and do strange and exciting …

…*et to be countable/uncountable. Prior experience with proofs and …*

…--11:55am

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

## …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: 54 (max 55)

# Teach For Splash

**Teach 2500+ high school students all that you know**

Jo
M
in
you

Past c
includ
Monac
My Very
Extreme
and mor

**Nov 21 and**
**Register Now**
**Deadline: Octo**
**Teach a 1 hr cla**

## S9802: How to make a Tesla Coil

Difficulty: ***

Teachers: peter Krogen

Ever wonder how a Tesla coil works? Interested in how you can make 10'+ sparks, and how those sparks can play music? Want to build your very own Tesla coil? Then this class is for you! We'll begin with a quick lecture on how a Tesla Coil works, from the bottom up. We'll then provide you with kits to build solid-state Tesla Coils that can make music, and we'll be there to assist you in building and debugging. The end product is a shoe box sized Tesla coil, capable of generating sparks and playing music.

**Prerequisites**
*It is recommended that students have a background working with electronics (ex, participated in FIRST) 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: 15 (max 16)

*Class temporarily full; please check back later*

## M11038: A Battle of Combinatorics Full!

Difficulty: **

Teachers: Luis Herrera Arias

Come ... about counting things you didn't know you could count. We'll play fun games ... crets of gambling.

**Meeting Time**
...un 9:05am--11:55am

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

## ...nting Beyond Infinity Full!

...lty: ****

Teachers: Dylan Hendrickson, Jordan Hines

...ting and never stopped? In this class, we'll talk about ordinals, the ...g this. We'll see many types of infinity and do strange and exciting

...*et to be countable/uncountable. Prior experience with proofs and*

...--11:55am

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

## ...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: 54 (max 55)

# Teach For Splash

**Teach 2500+ high school students all that you know**

Jo
M
in
you

Past c
includ
Monac
My Very
Extreme
and mor

**Nov 21 and**
**Register Now**
**Deadline: Octo**
**Teach a 1 hr cla**

## S9802: How to make a Tesla Coil

Difficulty: ***

Teachers: peter Krogen

Ever wonder how a Tesla coil works? Interested in how you can make 10'+ sparks, and how those sparks can play music? Want to build your very own Tesla coil? Then this class is for you! We'll begin with a quick lecture on how a Tesla Coil works, from the bottom up. We'll then provide you with kits to build solid-state Tesla Coils that can make music, and we'll be there to assist you in building and debugging. The end product is a shoe box sized Tesla coil, capable of generating sparks and playing music.

**Prerequisites**
*It is recommended that students have a background working with electronics (ex, participated in FIRST) 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: 15 (max 16)

*Class temporarily full; please check back later*

## M11038: A Battle of Combinatorics Full!

Difficulty: **

Teachers: Luis Herrera Arias

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

**eting Time**
un 9:05am--11:55am

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

## nting Beyond Infinity Full!

lty: ****

Teachers: Dylan Hendrickson, Jordan Hines

ting and never stopped? In this class, we'll talk about ordinals, the g this. We'll see many types of infinity and do strange and exciting

*et to be countable/uncountable. Prior experience with proofs and*

--11:55am

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

## 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: 54 (max 55)

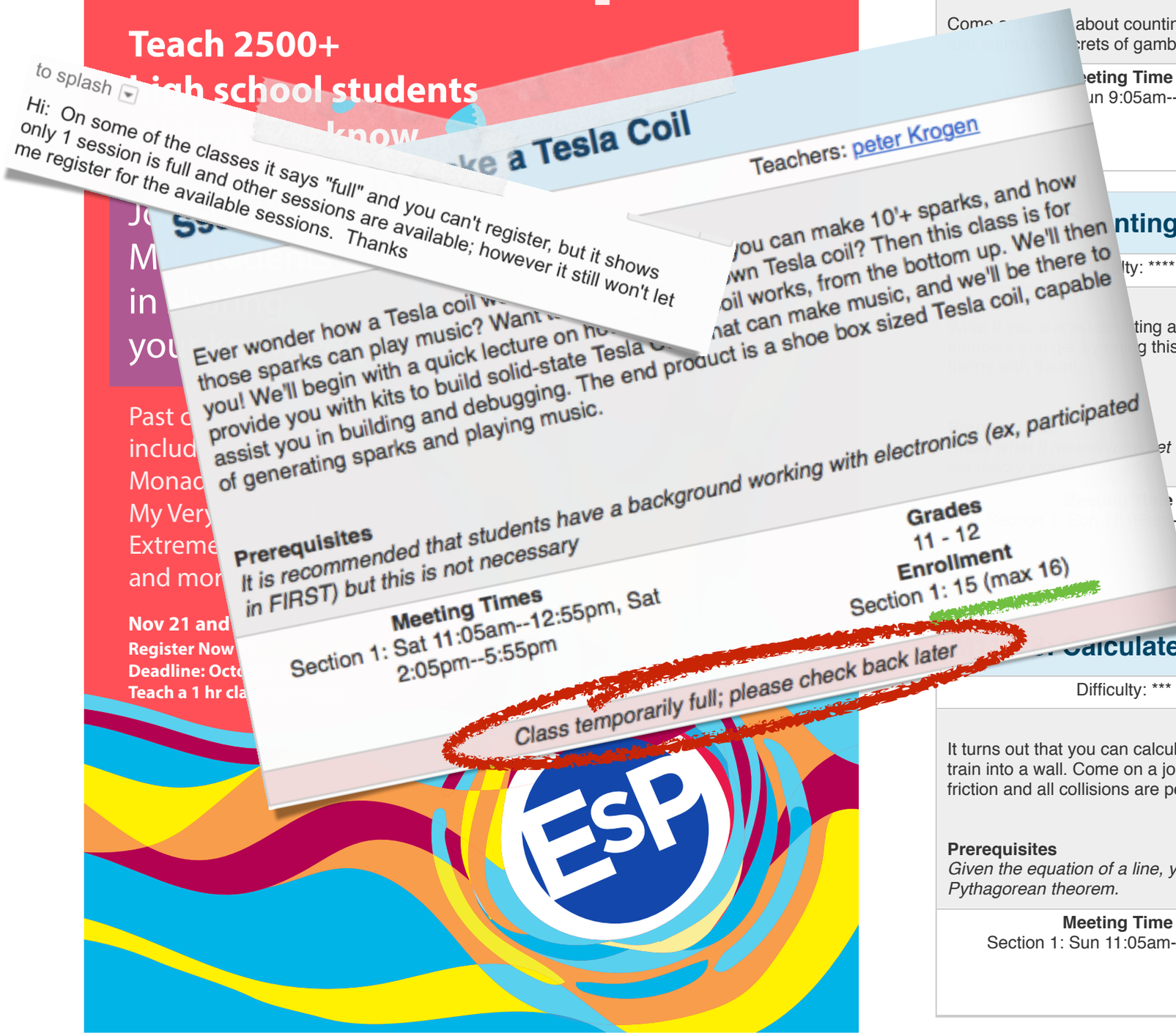
# Teach For Splash

**Teach 2500+**
**high school students**

Jo
M
in
you

Past c
includ
Monac
My Very
Extreme
and mor

**Nov 21 and**
**Register Now**
**Deadline: Octo**
**Teach a 1 hr cla**




## M11038: A Battle of Combinatorics Full!

Difficulty: ** | Teachers: Luis Herrera Arias

Come about counting things you didn't know you could count. We'll play fun games crets of gambling.

**eting Time**
un 9:05am--11:55am

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

## nting Beyond Infinity Full!

lty: **** | Teachers: Dylan Hendrickson, Jordan Hines

ting and never stopped? In this class, we'll talk about ordinals, the g this. We'll see many types of infinity and do strange and exciting

*et to be countable/uncountable. Prior experience with proofs and*

--11:55am

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

## 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: 54 (max 55)

# Teach For Splash

**Teach 2500+**
**high school students**

Past c
includ
Monac
My Very
Extreme
and mor

**Nov 21 and**
**Register Now**
**Deadline: Octo**
**Teach a 1 hr cla**

to splash

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

ke a Tesla Coil

Teachers: pete

Ever wonder how a Tesla coil w
those sparks can play music? Want
you! We'll begin with a quick lecture on ho
provide you with kits to build solid-state Tesla C
assist you in building and debugging. The end product is a shoe bo
of generating sparks and playing music.

**Prerequisites**
*It is recommended that students have a background working v*
*in FIRST) 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: 15 (max 16)

*Class temporarily full; please check back later*

## M11038: A Battle of Combinatorics **Full!**

Teachers: Luis Herrera Arias

## A10204: Math-y Beading

Difficulty: **

Teachers: Vivian Wang

Beads are pretty, but polyhedra are prettier. We'll learn to make buckyballs (a.k.a. truncated icosahedra for math folks or C60 for chem folks) out of beads and string. By the end of the class, you'll have your own shiny geometric trinket to keep! Depending on time and interest, we might learn to make other geometric things...A fractal dodecahedron? Polyhedral carbon nanotori? The possibilities are (almost) endless.

**Prerequisites**
*We'll be working with seed beads (which are pretty small), so a little finger dexterity and a lot of patience will go a long way!*

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

**Grades**
9 - 12
**Enrollment**
Section 1: 9 (max 10)
Section 2: **Full!** (max 10)

*Class temporarily full; please check back later*

## 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: 54 (max 55)

# Teach For Splash

**Teach 2500+**
**high school students**

Past c
includ
Monac
My Very
Extreme
and mor

**Nov 21 and**
**Register Now**
**Deadline: Octo**
**Teach a 1 hr cla**

to splash

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

to ESP

I have having trouble with changing classes, in that despite the enrollment number on the class being less than the max, it does not allow me to sing up because "class is temporarily full," I have removed my classes that I did have the period, and this class is only available in a single section (its A9946).

Ever wonder how a Tesla coil
those sparks can play music? Want
you! We'll begin with a quick lecture on ho
provide you with kits to build solid-state Tesla
assist you in building and debugging. The end product is a shoe bo
of generating sparks and playing music.

**Prerequisites**
*It is recommended that students have a background working w*
*in FIRST) 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: 15 (max 16)

*Class temporarily full; please check back later*

## M11038: A Battle of Combinatorics Full!

Teachers: Luis Herrera Arias

Teachers: Vivian Wang

*are pretty, but polyhedra are prettier. We'll learn to make buckyballs (a.k.a. truncated icosahedra for math folks or C60 for chem folks) out of beads and string. By the end of the class, you'll have your own shiny geometric trinket to keep! Depending on time and interest, we might learn to make other geometric things...A fractal dodecahedron? Polyhedral carbon nanotori? The possibilities are (almost) endless.*

**Prerequisites**
*We'll be working with seed beads (which are pretty small), so a little finger dexterity and a lot of patience will go a long way!*

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

**Grades**
9 - 12
**Enrollment**
Section 1: 9 (max 10)
Section 2: **Full!** (max 10)

*Class temporarily full; please check back later*

## Calculate Pi with Trains!

Difficulty: ***

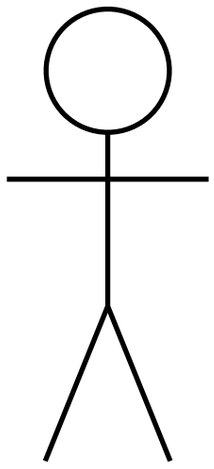
Teachers: Ziv Scully

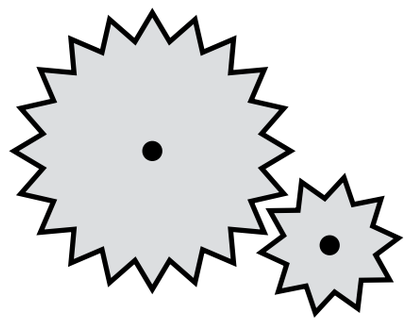
It turns out that you can calculate pi to very high accuracy by bouncing a small train and a big train into a wall. Come on a journey through Extra-Nice Physics Land (where there's no friction and all collisions are perfectly elastic) to see how it works!

**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: 54 (max 55)



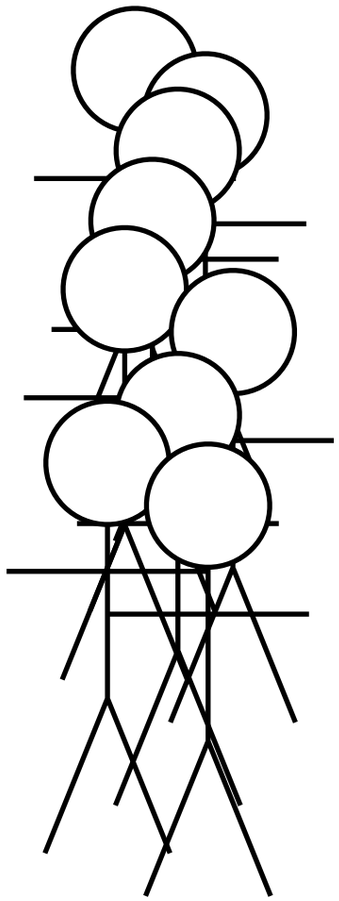


Web Server

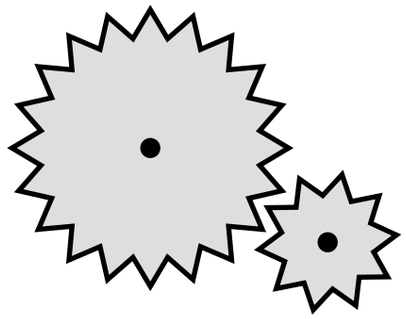
Request logic

Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |



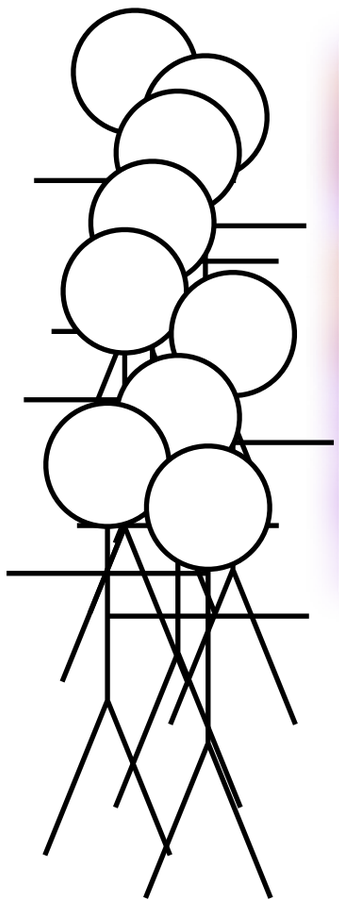
Web Server

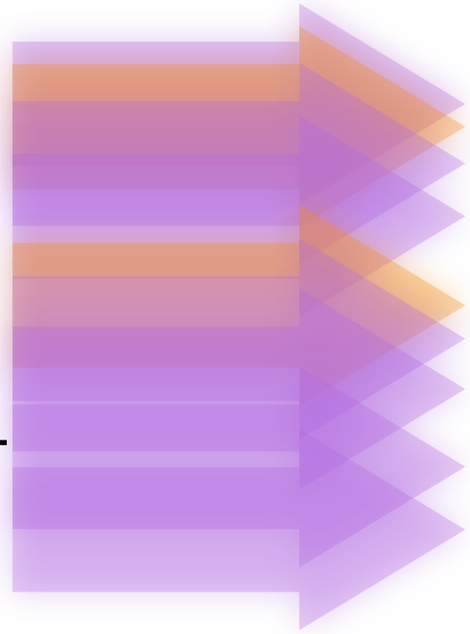


Request logic

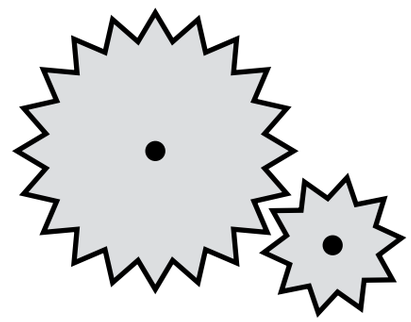
Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |

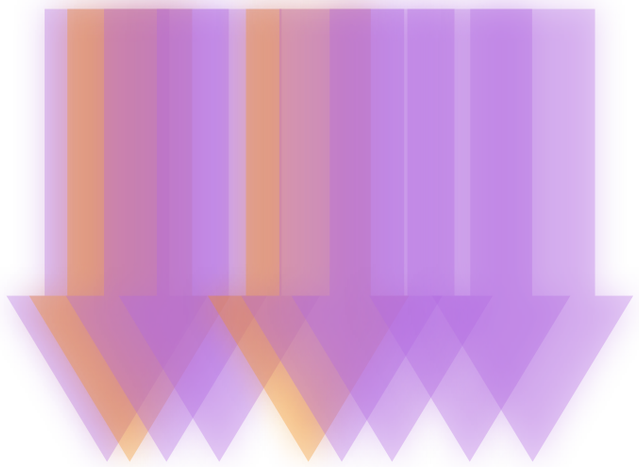




Web Server

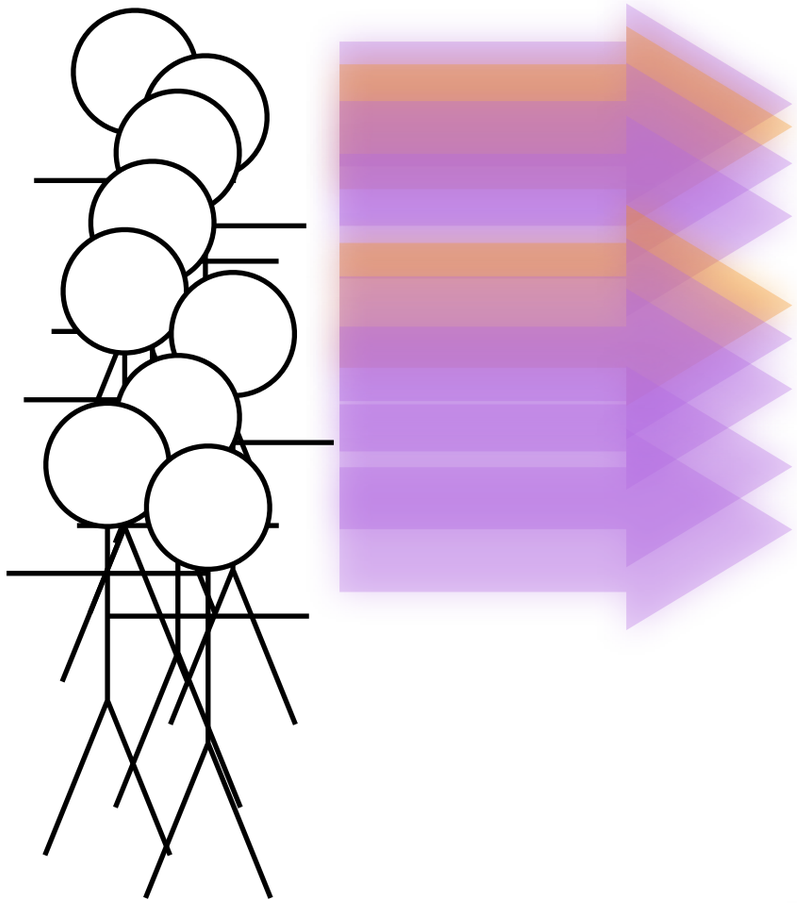


Request logic



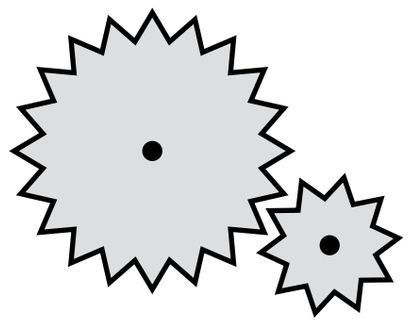
Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |





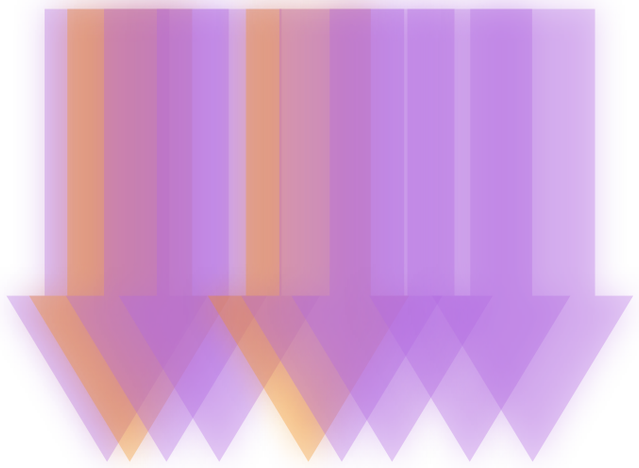
## Web Server



Request logic

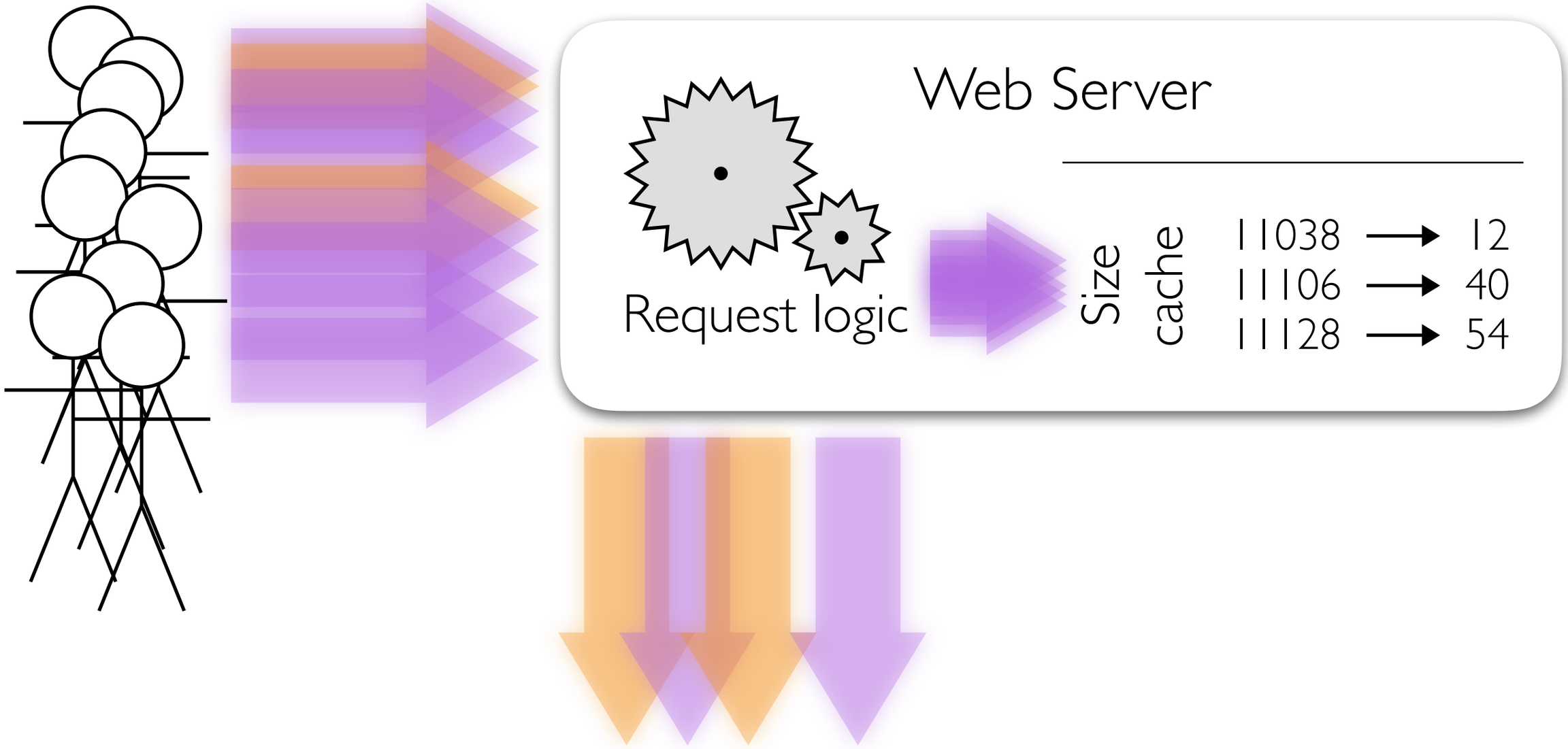
Size cache

| | |
|---|---|
| 11038 | 12 |
| 11106 | 40 |
| 11128 | 54 |



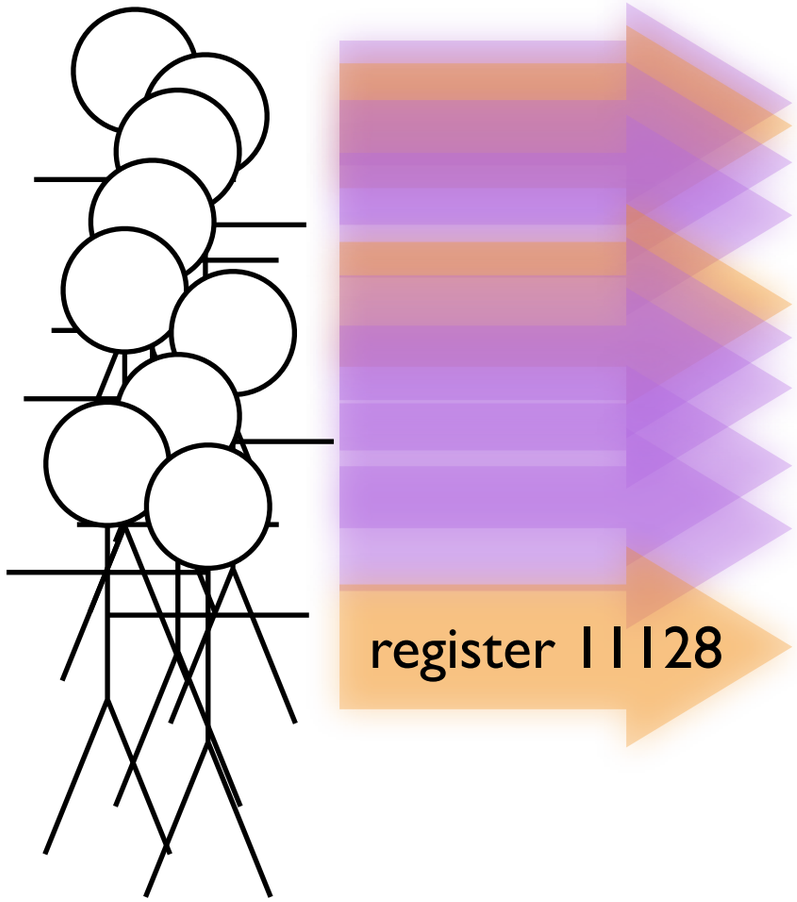
Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |



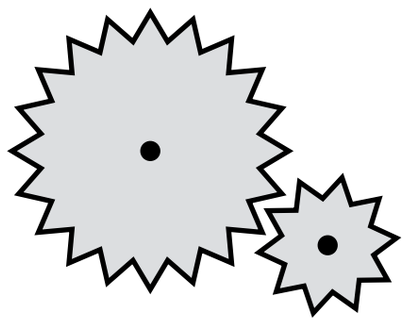

Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |



register 11128

Web Server

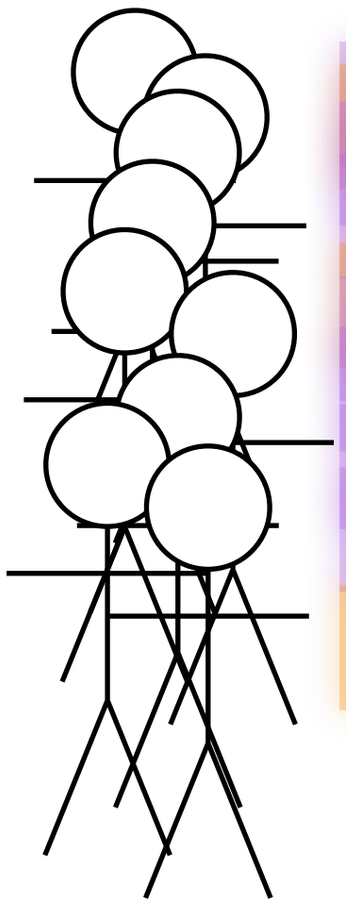


Request logic

| Size | cache |
|---|---|
| 11038 | → 12 |
| 11106 | → 40 |
| 11128 | → 54 |

Database

| *id* | *title* | *max_size* | *size* |
|---|---|---|---|
| 11038 | "A Battle of Combinatorics" | 12 | 12 |
| 11106 | "Counting Beyond Infinity" | 40 | 40 |
| 11128 | "Calculate Pi With Trains!" | 55 | 54 |



register 11128

Web Server

Request logic

Size cache

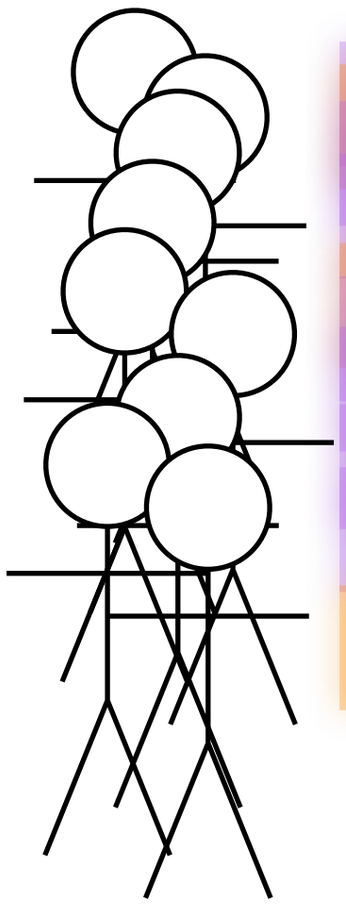
| | |
|---|---|
| 11038 | 12 |
| 11106 | 40 |
| 11128 | 54 |

update

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

Database

| *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** |



register 11128

Web Server

Request logic

Size cache

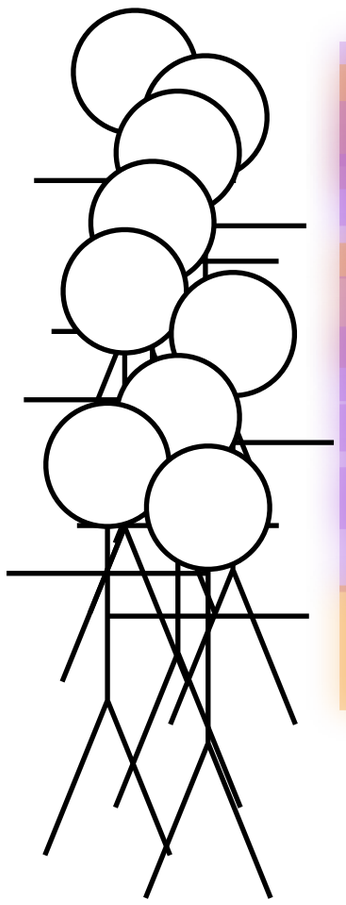
| | |
|---|---|
| 11038 | 12 |
| 11106 | 40 |
| 11128 | ~~54~~ |

update

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

Database

| *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** |



register 11128

Web Server

Request logic

Program instrumentation

Size cache

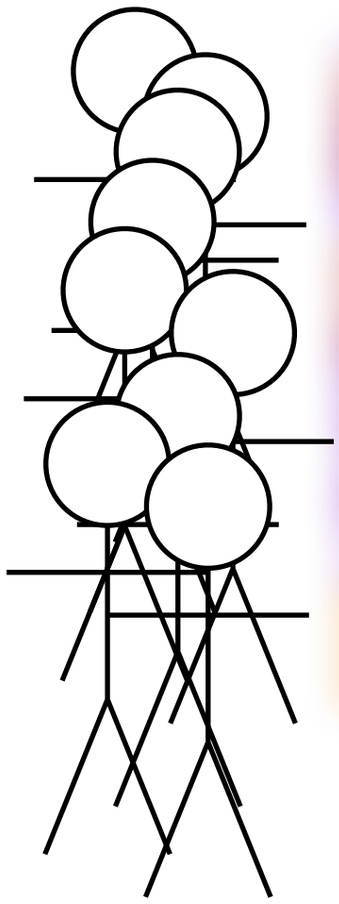
| | |
|---|---|
| 11038 | 12 |
| 11106 | 40 |
| 11128 | ~~54~~ |

update

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

Database

| *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** |



Web Server

Request logic

Program instrumentation

Size cache

11038 → 12
11106 → 40
11128 → ~~54~~

register 11128

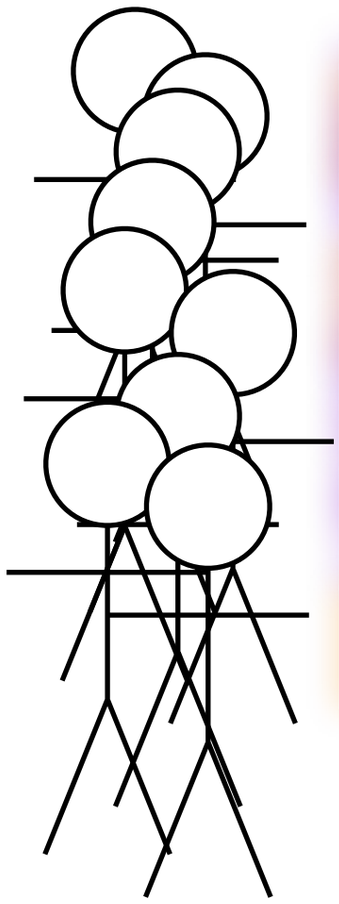
update

UPDATE *size* = *size* + 1
WHER

SQL analysis

Database

| *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** |



Web Server

Request logic

Size

Cache

11038 → 12

11106 → 40

Program instrumentation

Cache data structure

register 11128

update

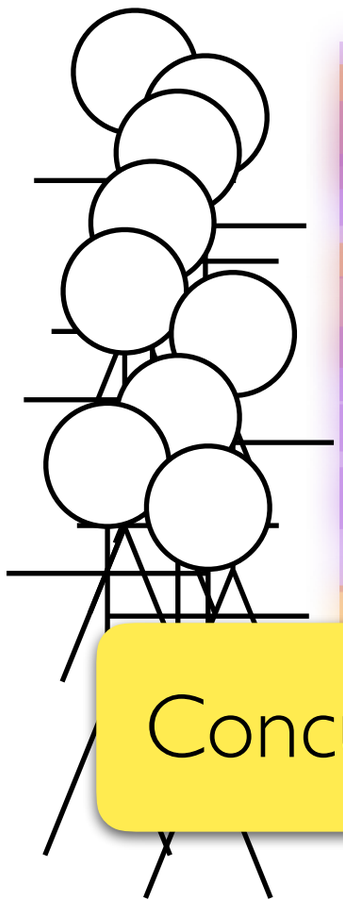
UPDATE *size* = *size* + 1

WHER

SQL analysis

Database

| *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** |



Web Server

Request logic

Size

Cache

11038 → 12

11106 → 40

11128 → 55

Program instrumentation

Cache data structure

Concurrency control

update

UPDATE *size* = *size* + 1

WHER

SQL analysis

Database

| *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** |

# Approaches to Caching

| | Caching | Automatic | Flexible |
|---|---|---|---|
| No caching | ✗ | ✓ | ✓ |
| Manual instrumentation | ✓ | ✗ | ✓ |
| Library (e.g. ORM) | ✓ | ✓ | ✗ |

# Approaches to Caching

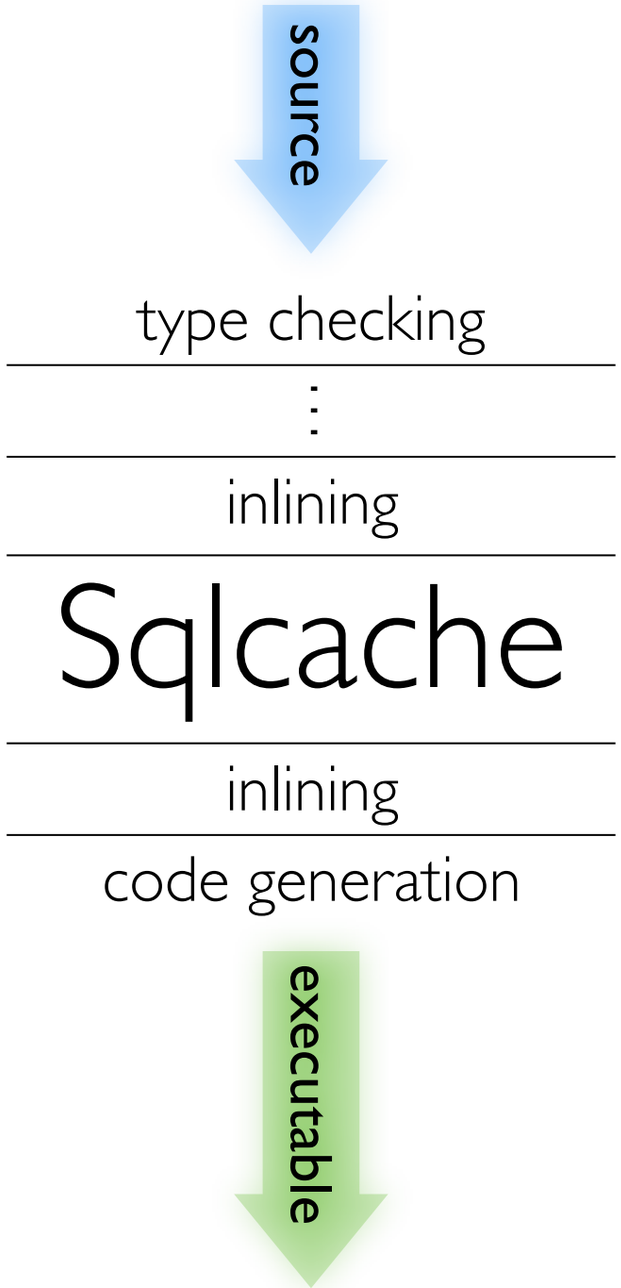
| | Caching | Automatic | Flexible |
|---|---|---|---|
| No caching | ✗ | ✓ | ✓ |
| Manual instrumentation | ✓ | ✗ | ✓ |
| Library (e.g. ORM) | ✓ | ✓ | ✗ |
| Compiler optimization | ✓ | ✓ | ✓ |

# Sqlcache

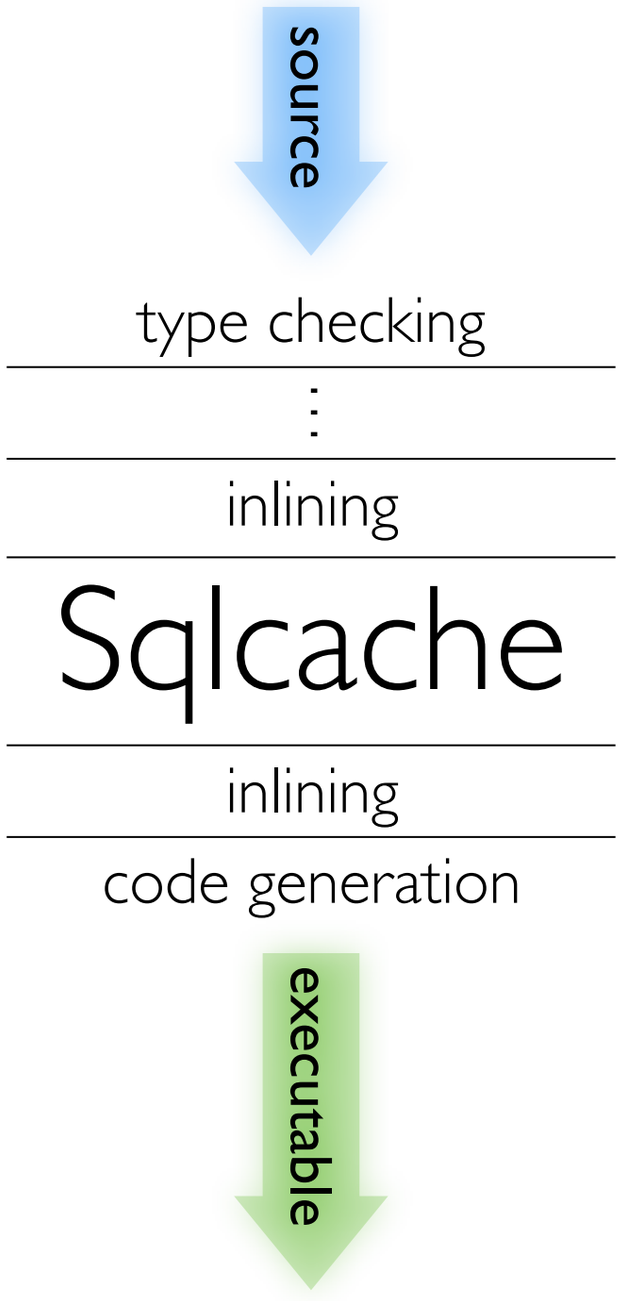
a compiler optimization for caching in the

Ur/Web programming language

# Ur/Web compiler



# Ur/Web compiler




Program instrumentation

SQL analysis

Cache data structure

Concurrency control

*all automatically*

for single-server applications

# 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

```
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>
```

cached region

# 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>
```

cached region

# 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>
```

cached region

cache invalidations

# 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>
```

cached region

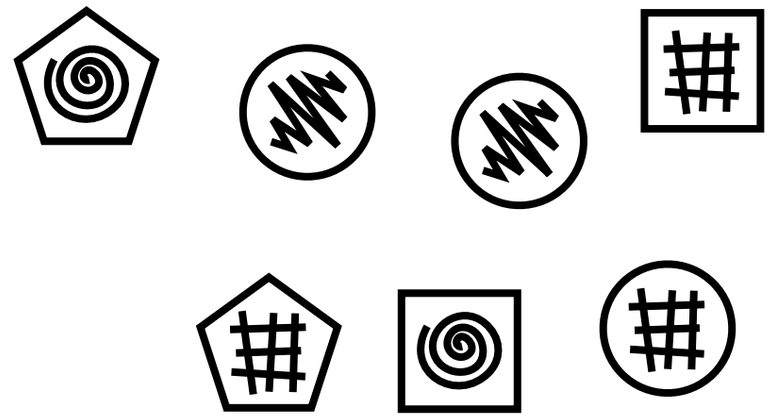
cache invalidations

# Invalidation for INSERT

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

Cache

Database



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

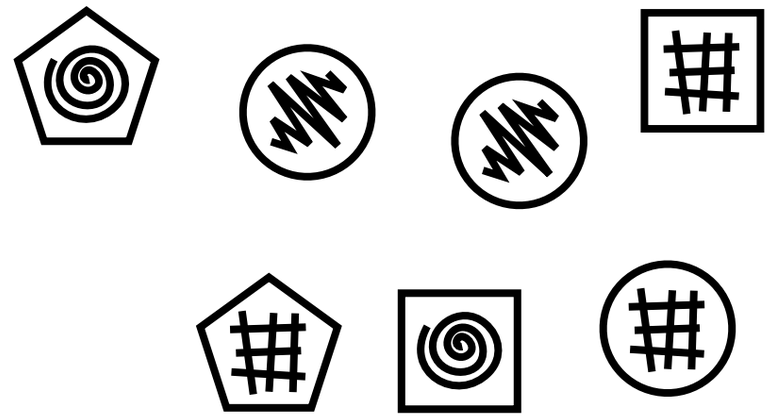
# Invalidation for INSERT

SELECT *shape* WHERE *fill* = $\mathbf{x}$

Cache

$\mathbf{x}$ =

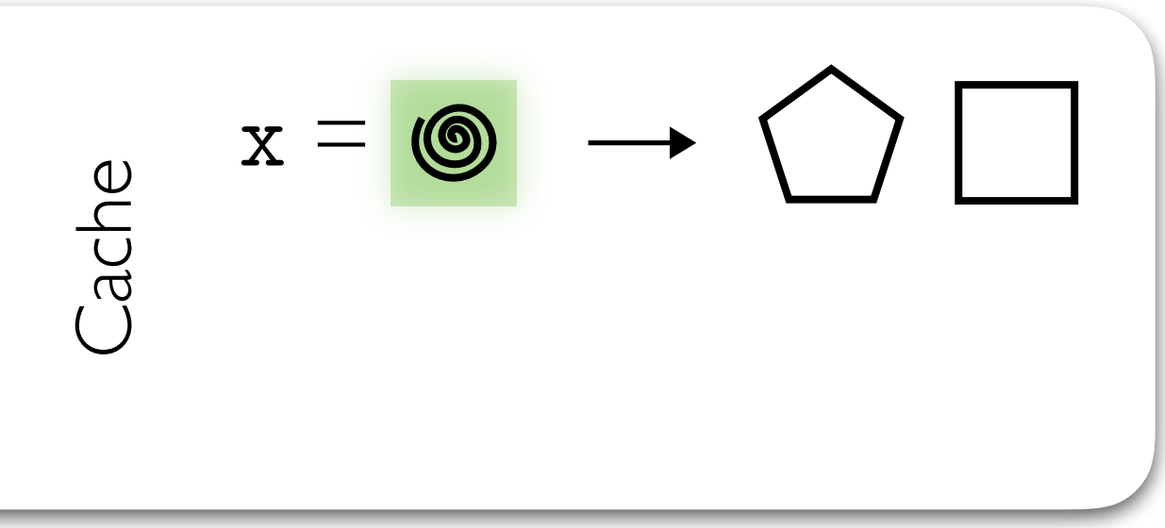
Database

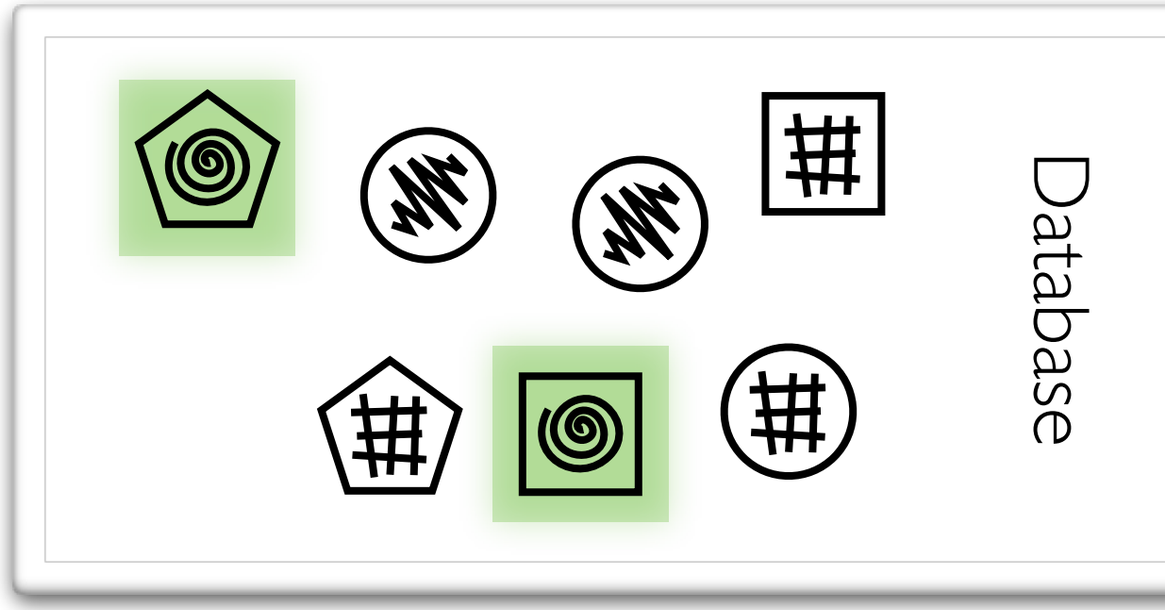


INSERT (*shape*, *fill*) = $(\mathbf{y}, \mathbf{z})$

# Invalidation for INSERT

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

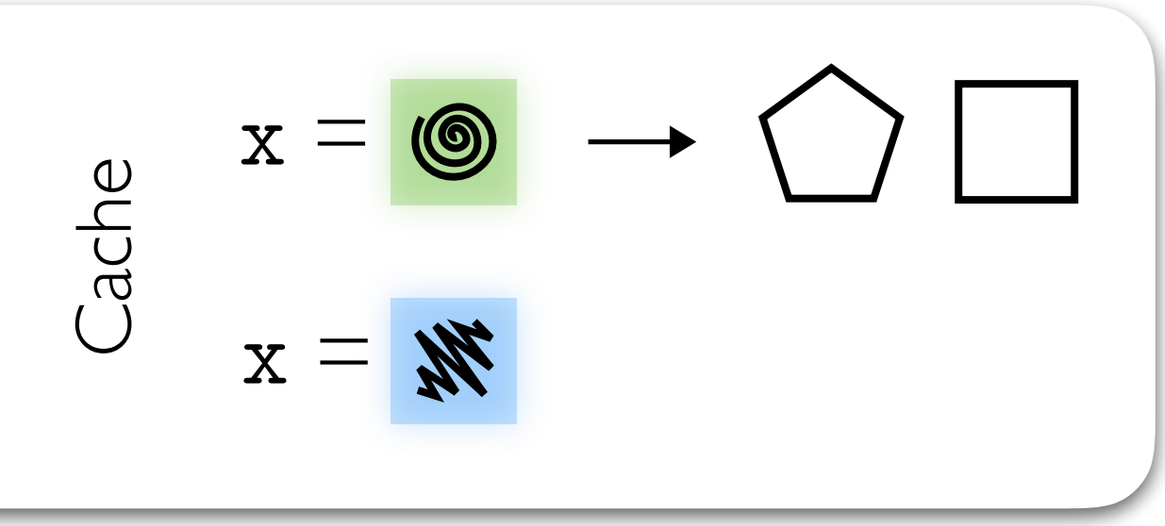


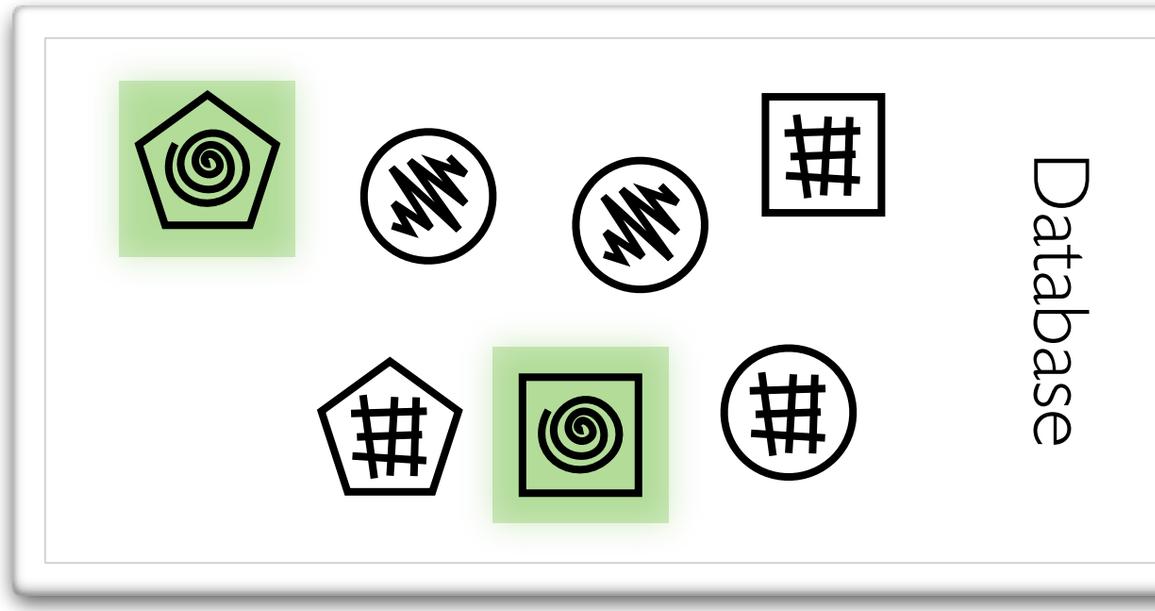




INSERT (*shape*, *fill*) = (`y`, `z`)

# Invalidation for INSERT

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

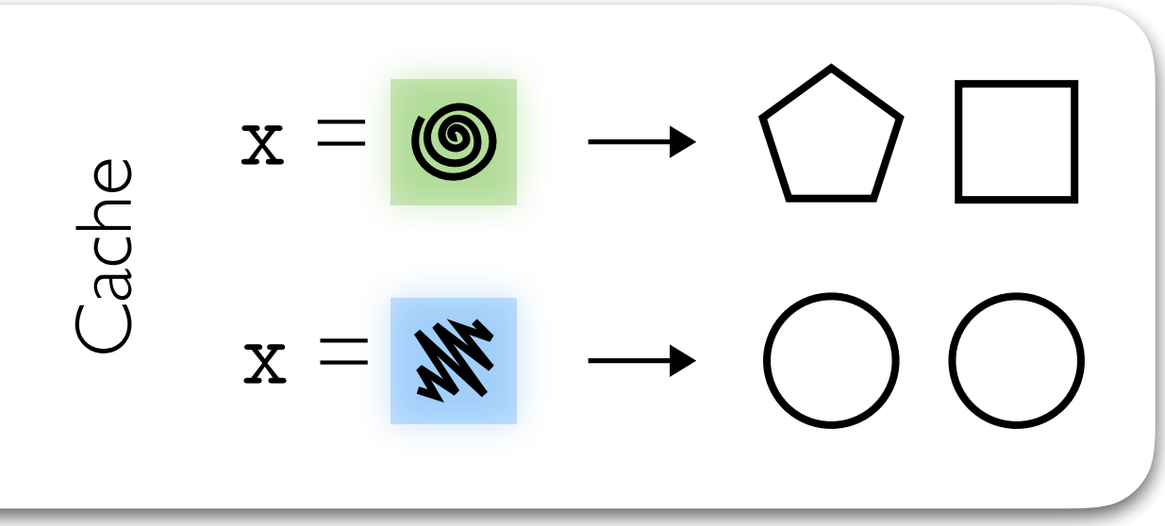


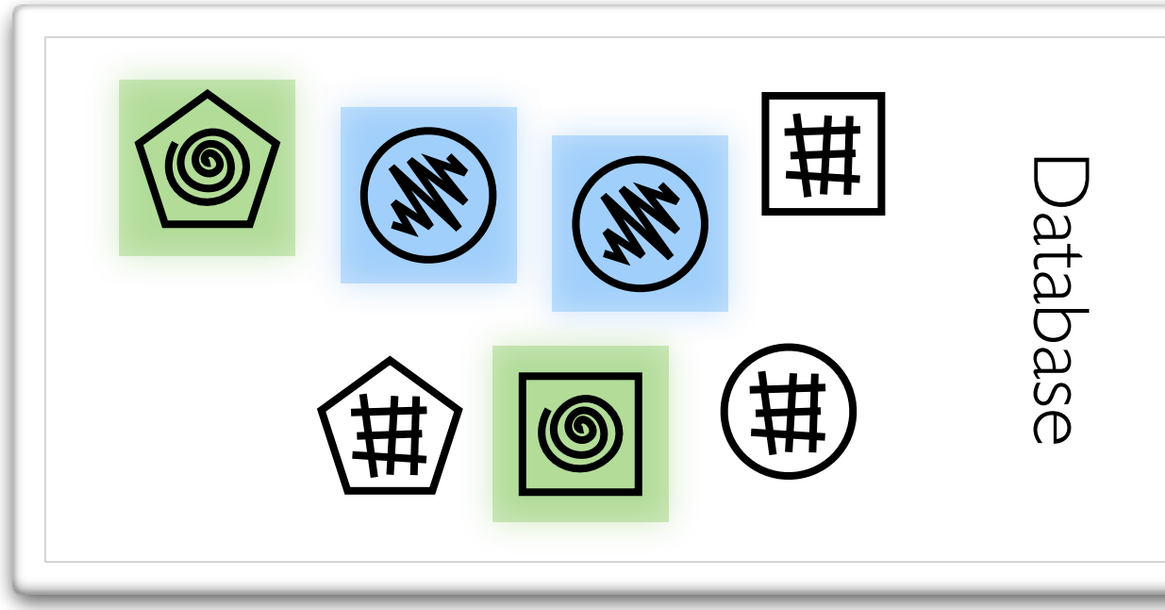




INSERT (*shape*, *fill*) = (`y`, `z`)

# Invalidation for INSERT

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

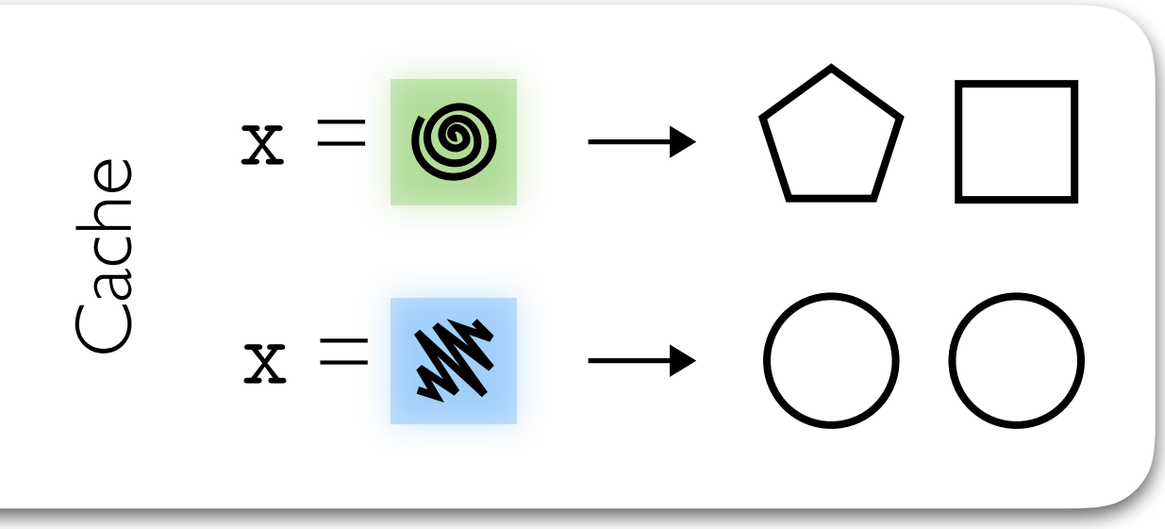


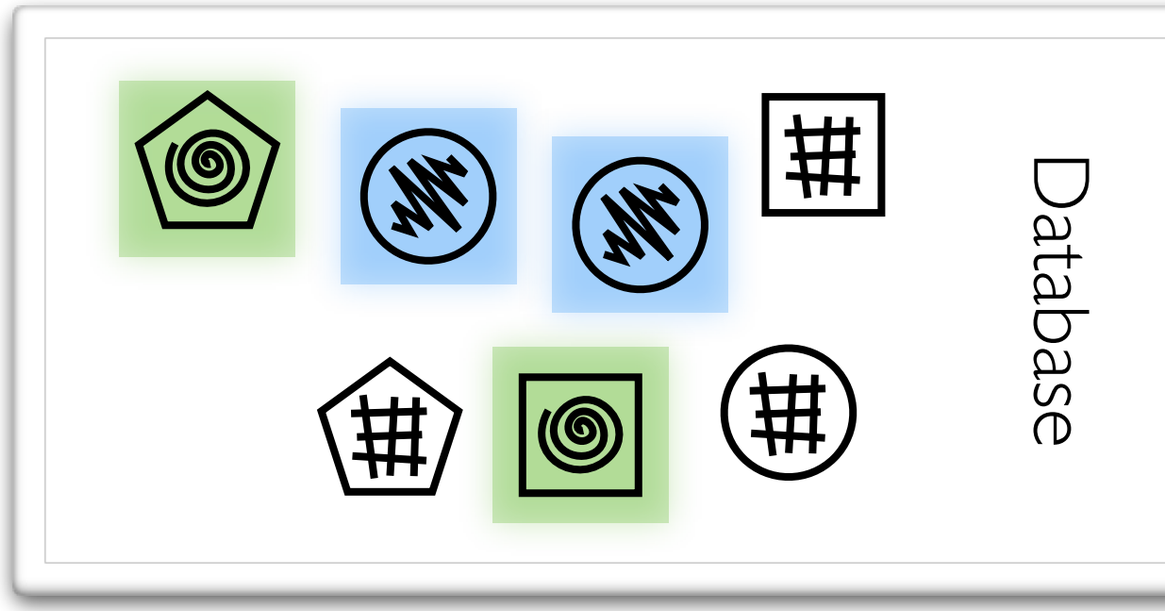




INSERT (*shape*, *fill*) = (`y`, `z`)

# Invalidation for INSERT

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



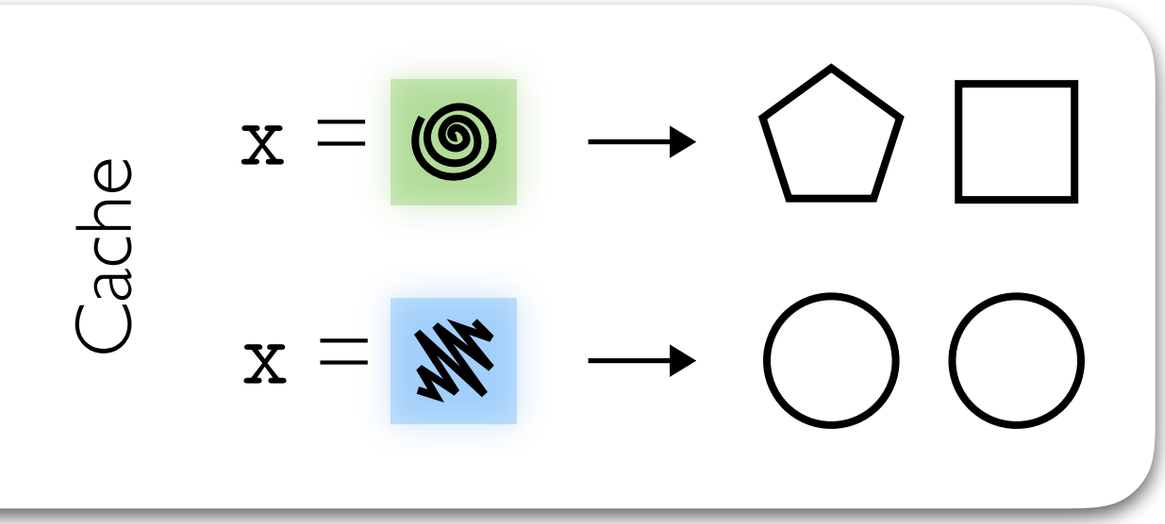



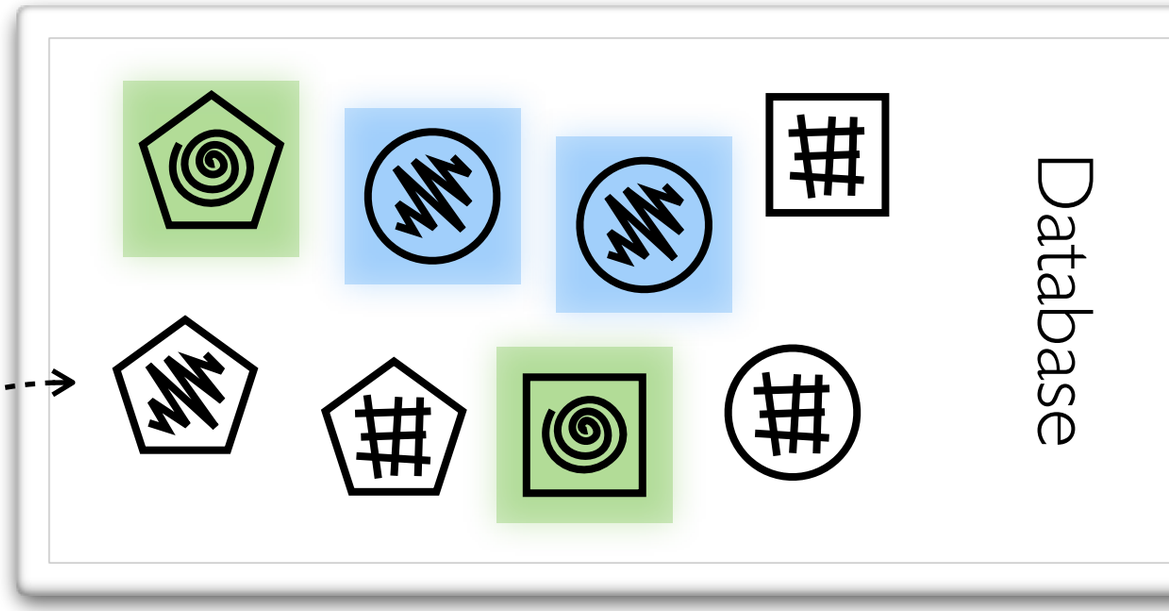

INSERT (*shape*, *fill*) = (`y`, `z`)

(`y`, `z`) =

# Invalidation for INSERT

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



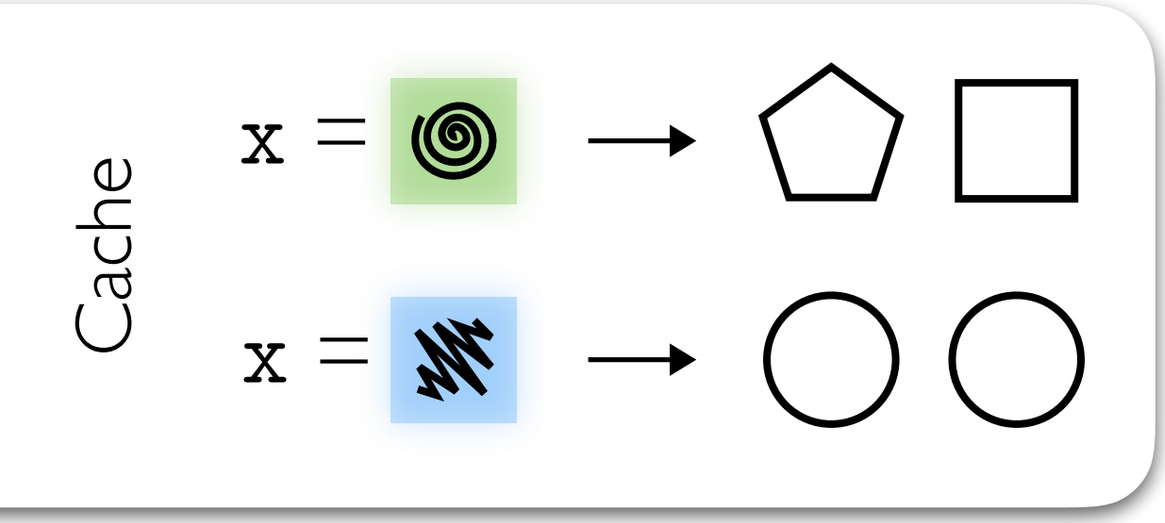



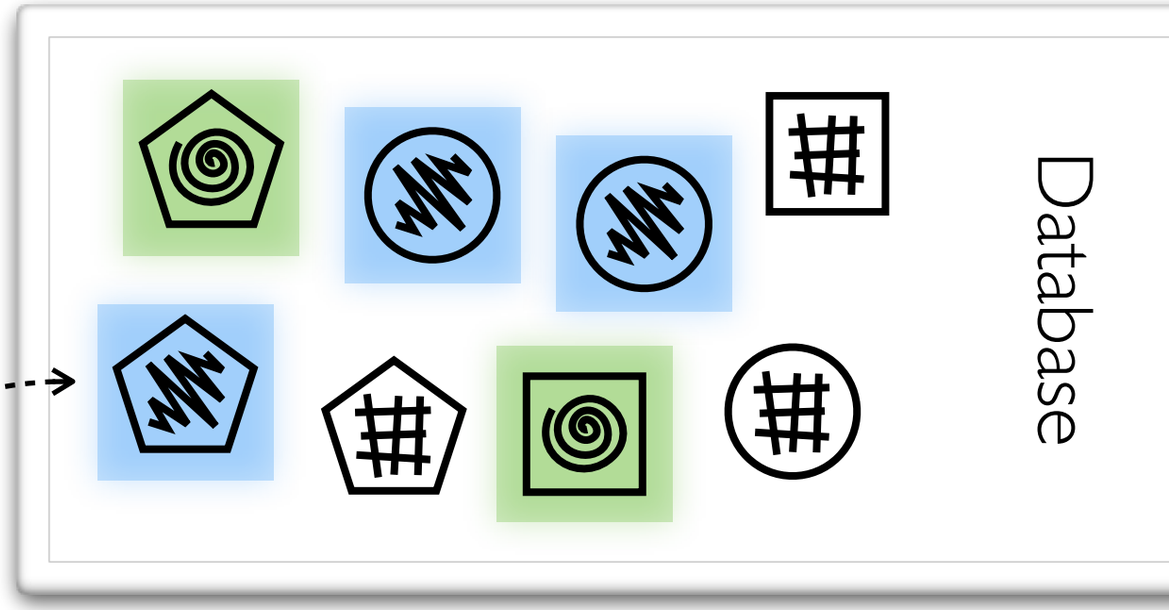

INSERT (*shape*, *fill*) = (`y`, `z`)

(`y`, `z`) =

# Invalidation for INSERT

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



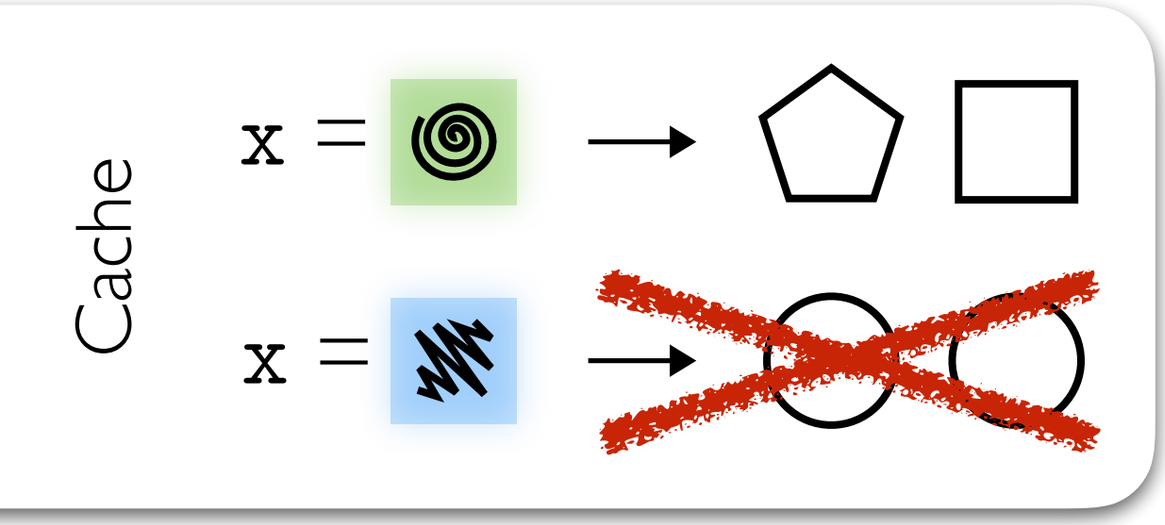



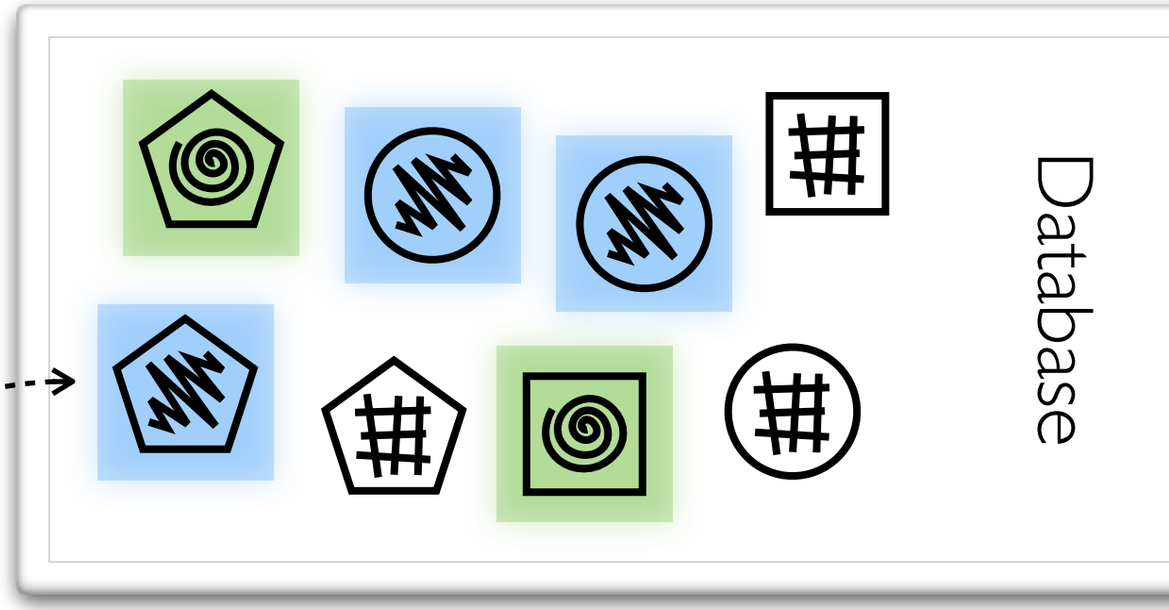

INSERT (*shape*, *fill*) = (`y`, `z`)

(`y`, `z`) =

# Invalidation for INSERT

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



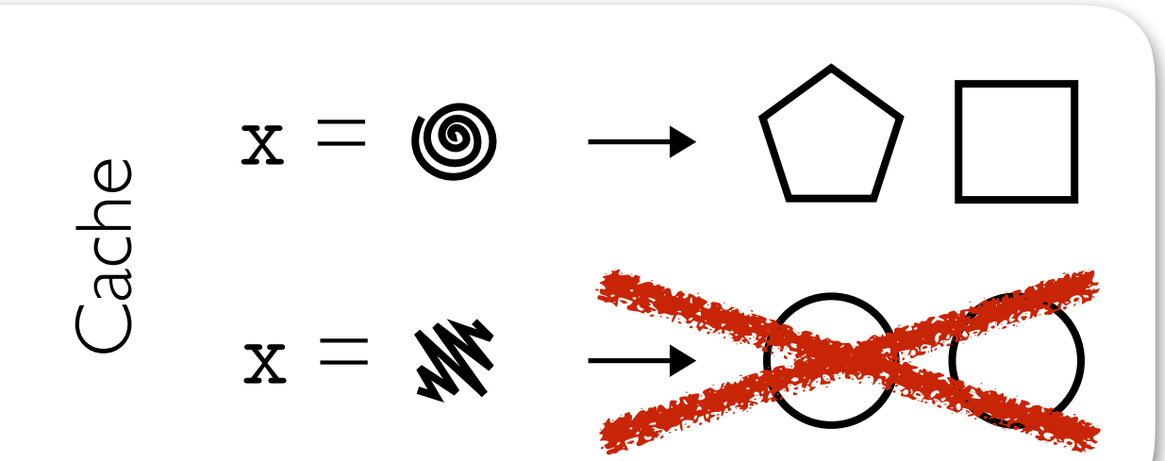



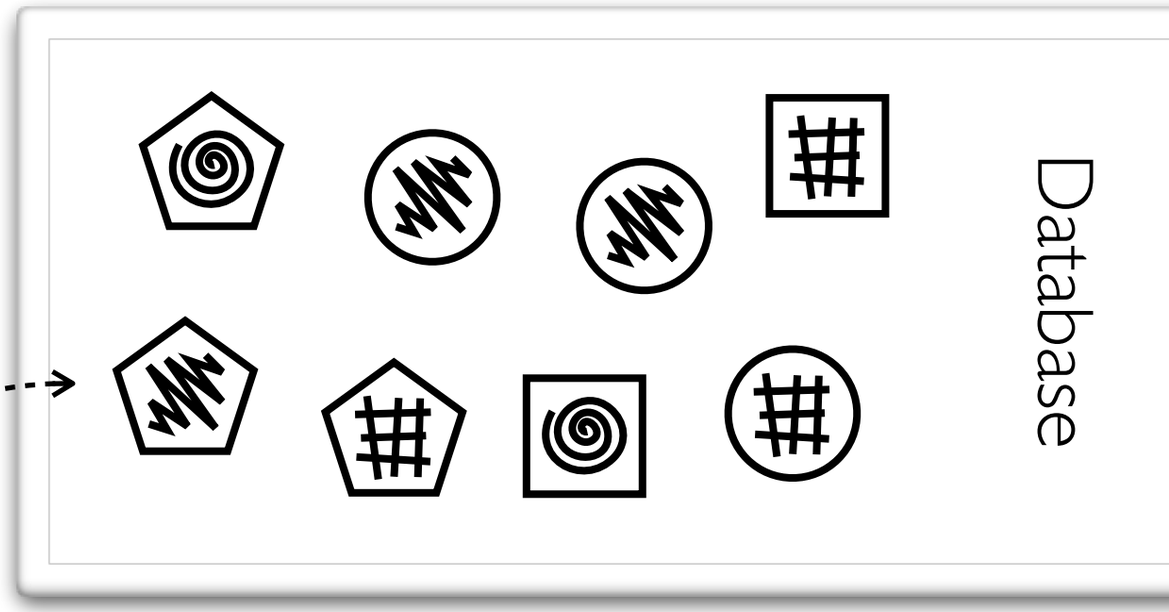

INSERT (*shape*, *fill*) = (`y`, `z`)

(`y`, `z`) =

# Invalidation for INSERT

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



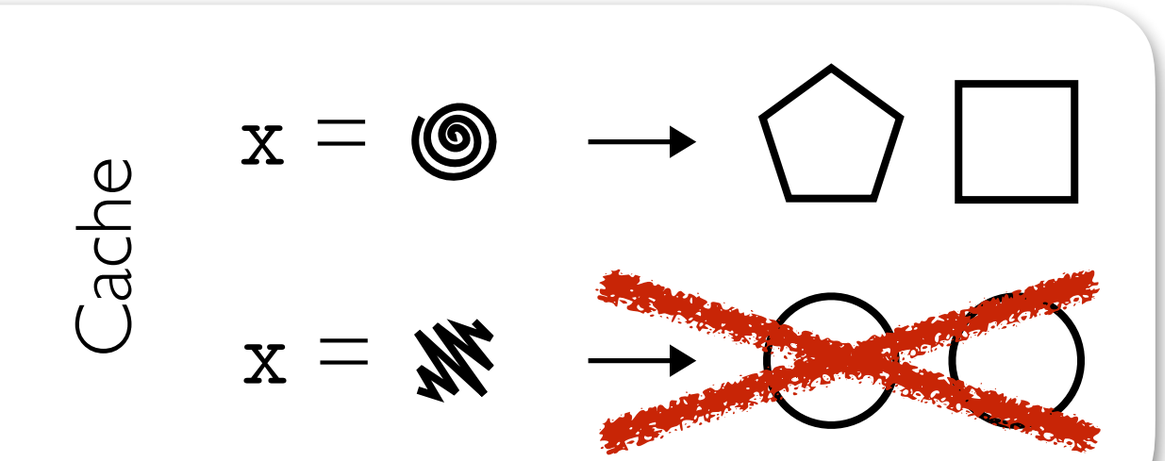



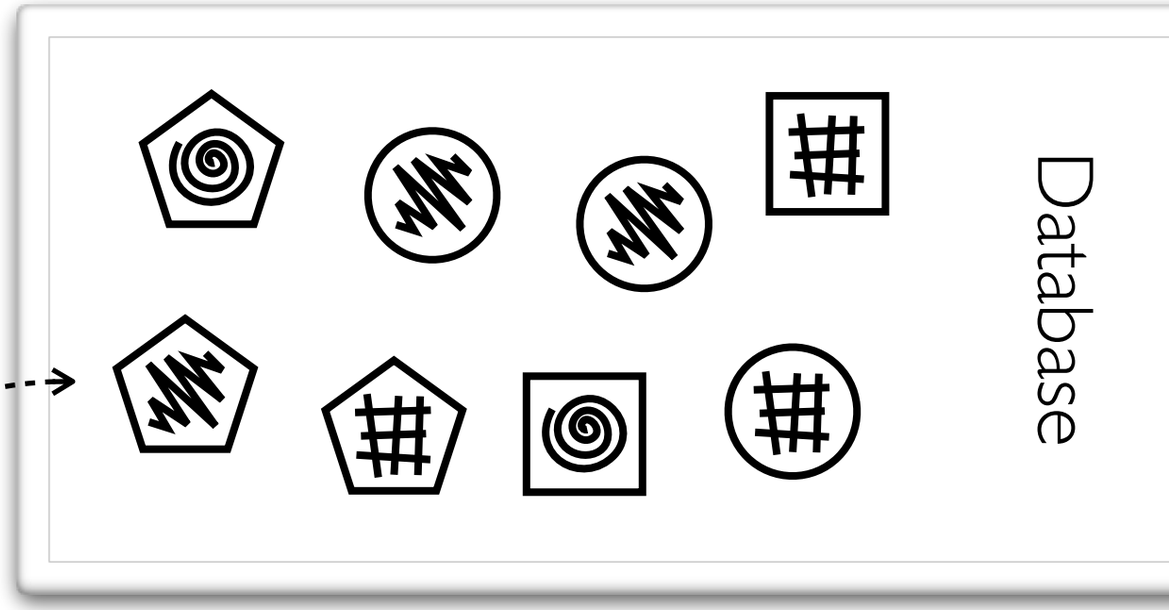

INSERT (*shape*, *fill*) = (`y`, `z`)

(`y`, `z`) =

# Invalidation for INSERT

SELECT *shape* WHERE *fill* = $\mathbf{x}$




INSERT (*shape*, *fill*) = ($\mathbf{y}$, $\mathbf{z}$)

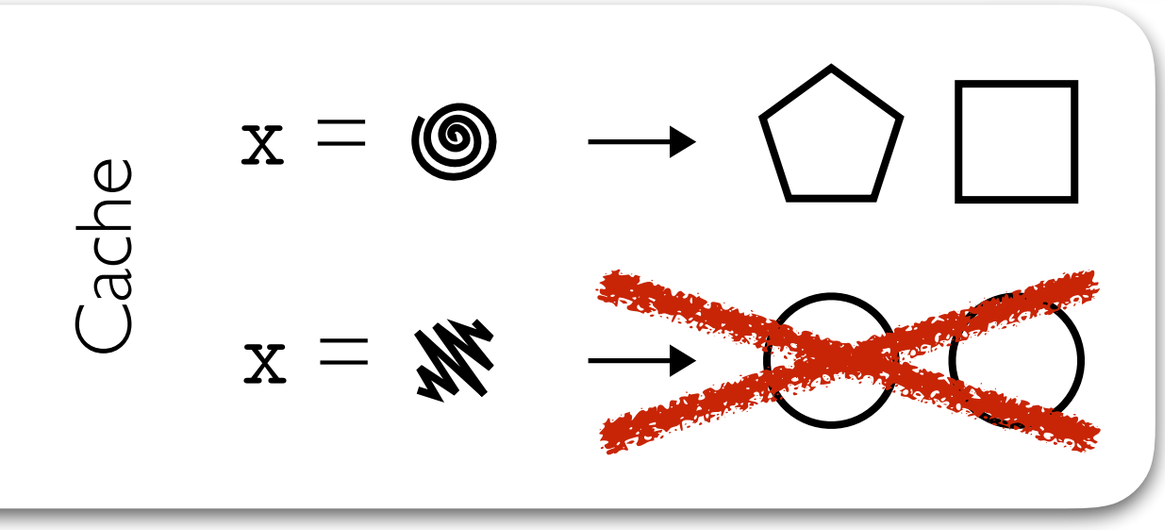
($\mathbf{y}$, $\mathbf{z}$) =

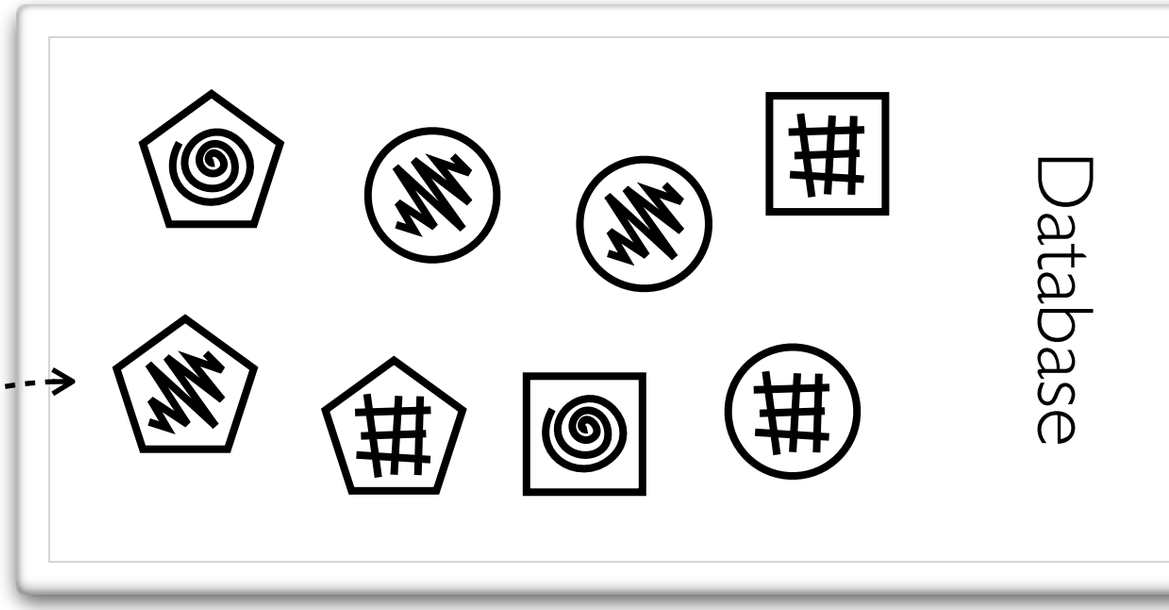
**Invalidation formula:**

$\exists$ (*shape*, *fill*).
*fill* = $\mathbf{x}$ $\wedge$ *shape* = $\mathbf{y}$ $\wedge$ *fill* = $\mathbf{z}$

# Invalidation for INSERT

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




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

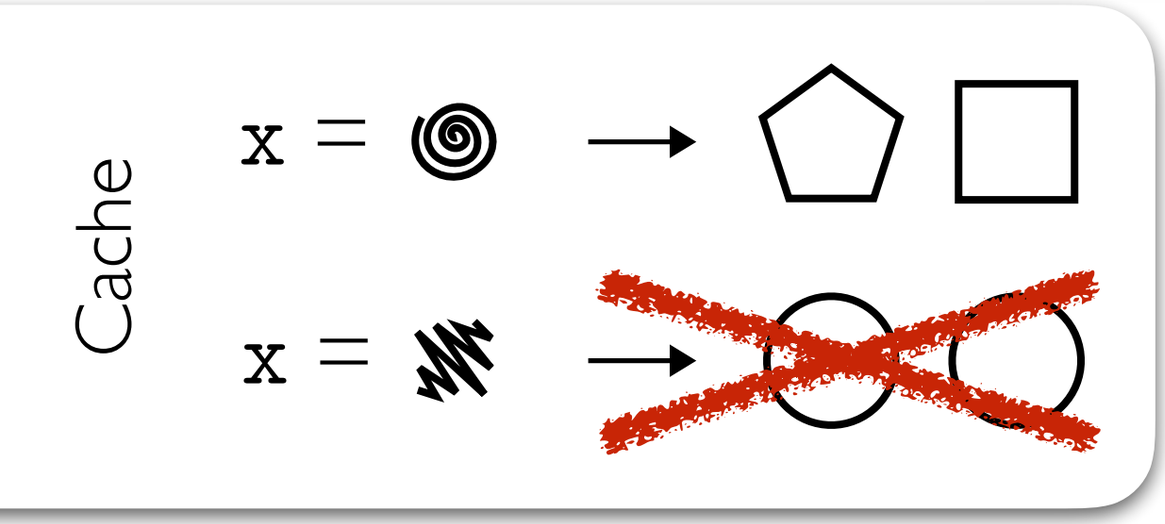
(`y`, `z`) =

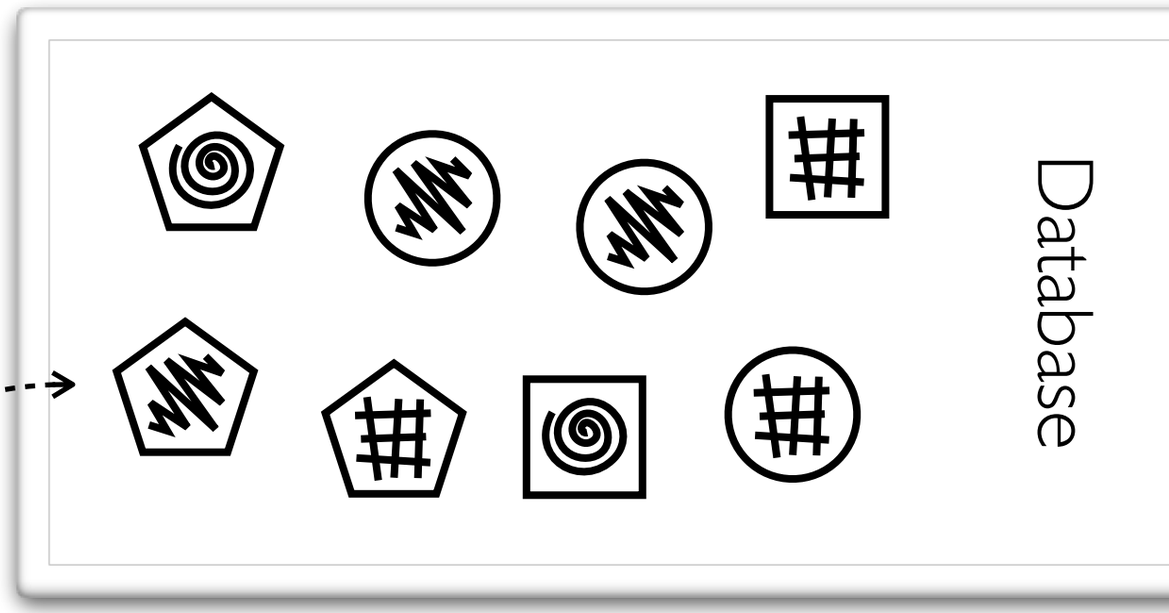
**Invalidation formula:**

∃ (*shape*, *fill*).
*fill* = `x` ∧ *shape* = `y` ∧ *fill* = `z`

# Invalidation for INSERT

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




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

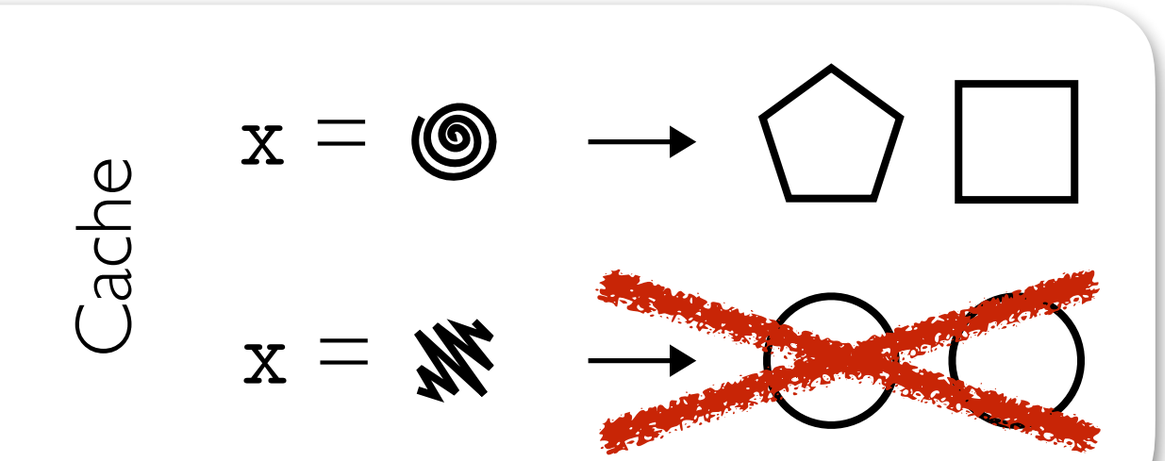
(`y`, `z`) =

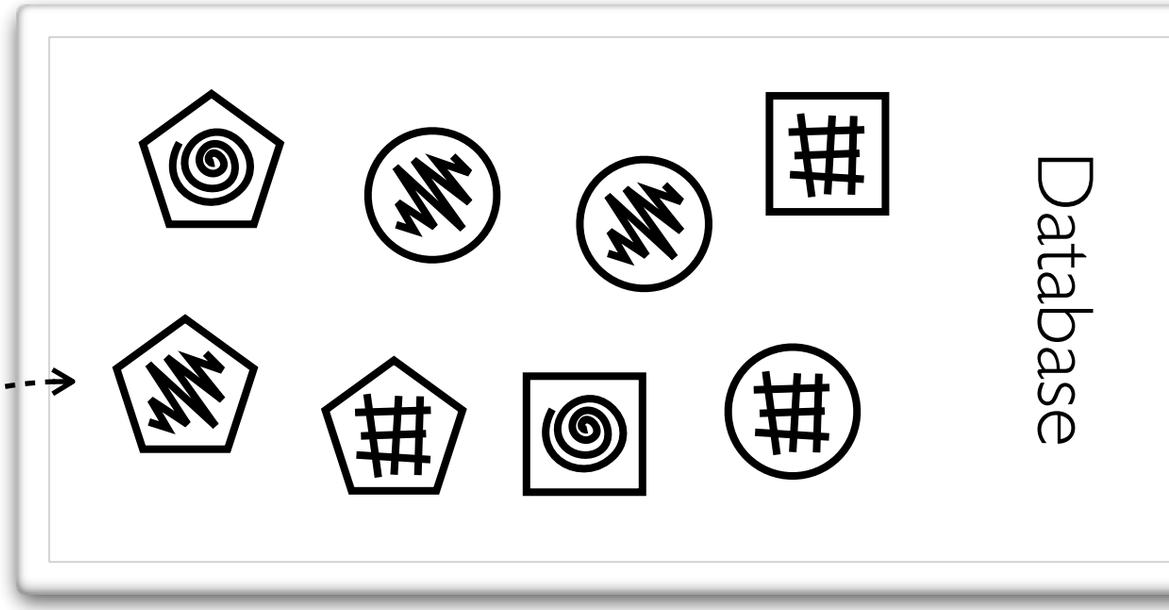
**Invalidation formula:**

∃ (*shape*, *fill*).
*fill* = `x` ∧ *shape* = `y` ∧ *fill* = `z`

# Invalidation for INSERT

SELECT *shape* WHERE *fill* = $\mathbf{x}$




INSERT (*shape*, *fill*) = ($\mathbf{y}$, $\mathbf{z}$)

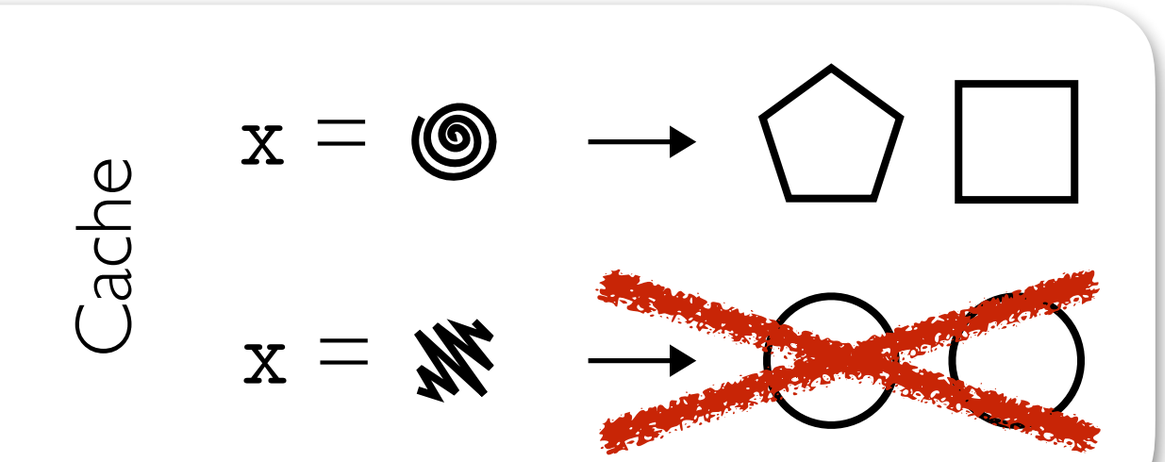
($\mathbf{y}$, $\mathbf{z}$) =

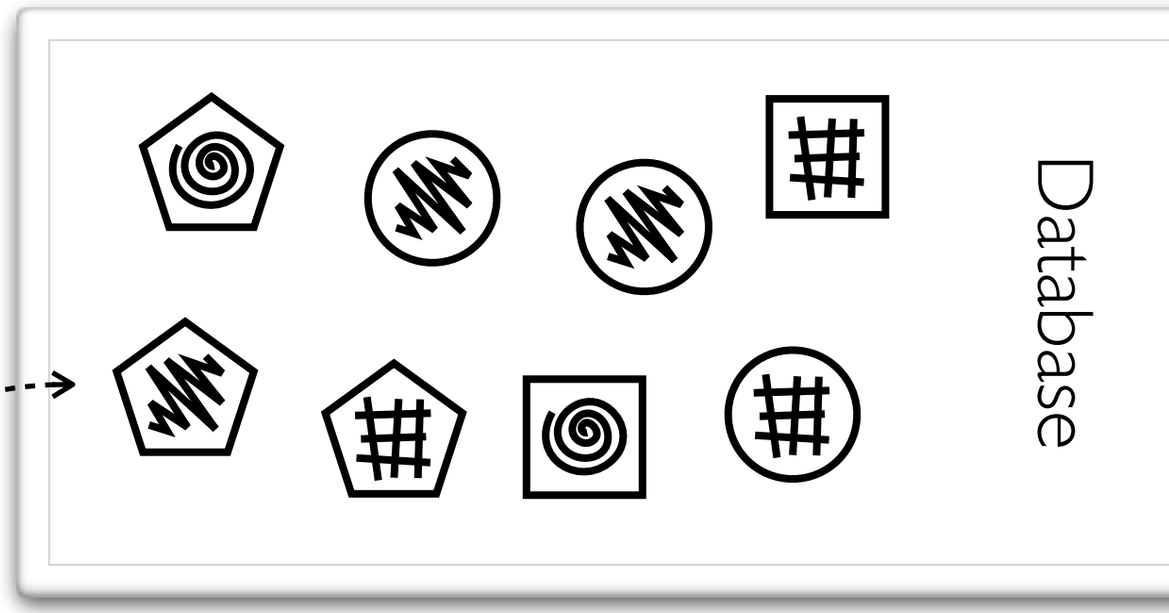
**Invalidation formula:**

$\exists$ (*shape*, *fill*).
*fill* = $\mathbf{x}$ $\wedge$ *shape* = $\mathbf{y}$ $\wedge$ *fill* = $\mathbf{z}$

# Invalidation for INSERT

SELECT *shape* WHERE *fill* = $\mathbf{x}$




INSERT (*shape*, *fill*) = ($\mathbf{y}$, $\mathbf{z}$)

($\mathbf{y}$, $\mathbf{z}$) =

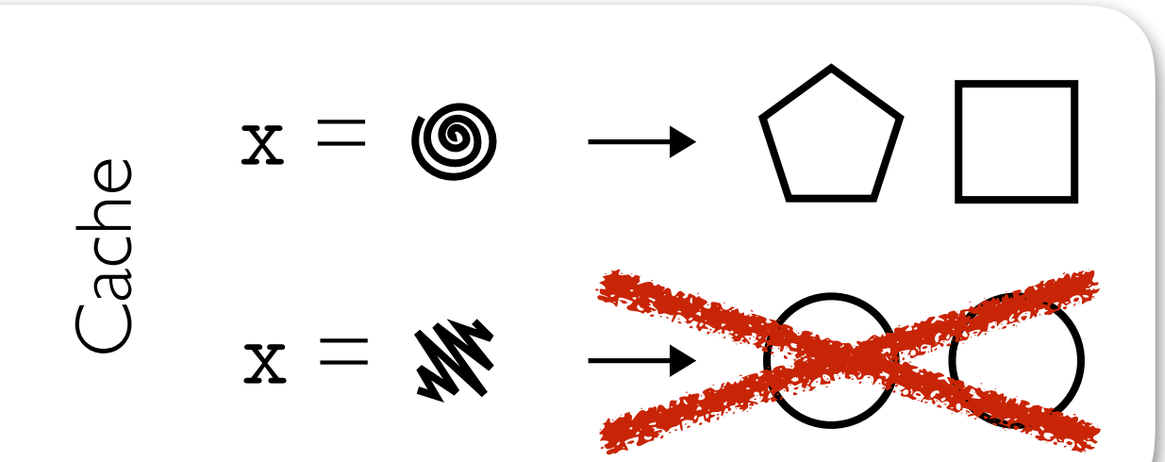
**Invalidation formula:**

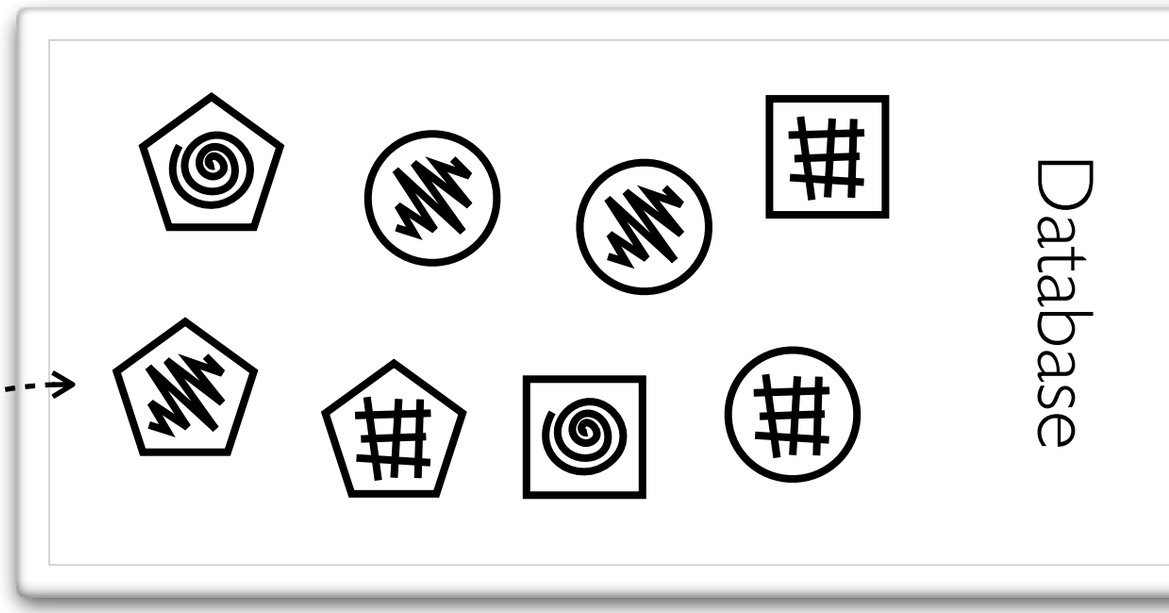
$\exists$ (*shape*, *fill*).
*fill* = $\mathbf{x}$ $\wedge$ *shape* = $\mathbf{y}$ $\wedge$ *fill* = $\mathbf{z}$

$\Longrightarrow$ $\mathbf{x} = \mathbf{z}$

# Invalidation for INSERT

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




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

(`y`, `z`) =

**Invalidation formula:**

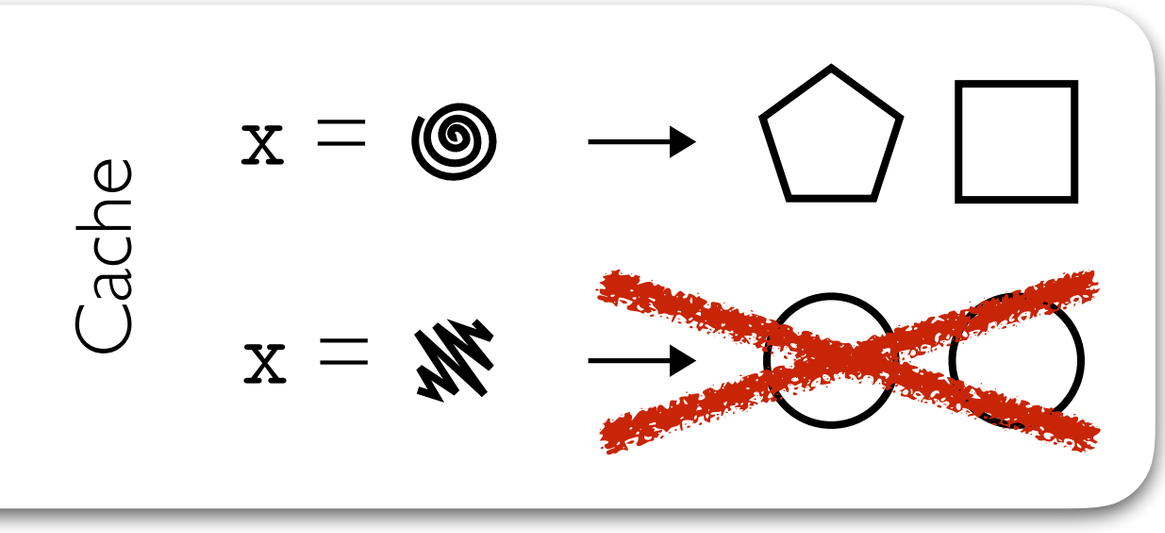
$\exists$ (*shape*, *fill*).
*fill* = `x` $\wedge$ *shape* = `y` $\wedge$ *fill* = `z`

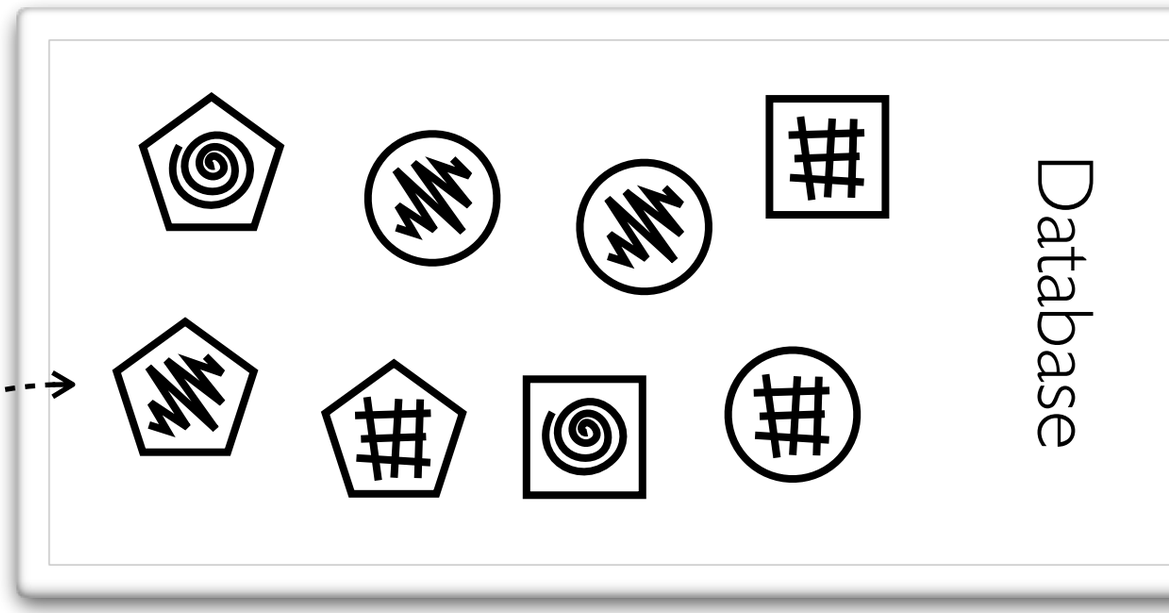
$\Rightarrow$ `x` = `z`

cache key

# Invalidation for INSERT

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




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

(`y`, `z`) =

**Invalidation formula:**

$$\exists\ (shape, fill).\ fill = \mathtt{x} \wedge shape = \mathtt{y} \wedge fill = \mathtt{z}$$

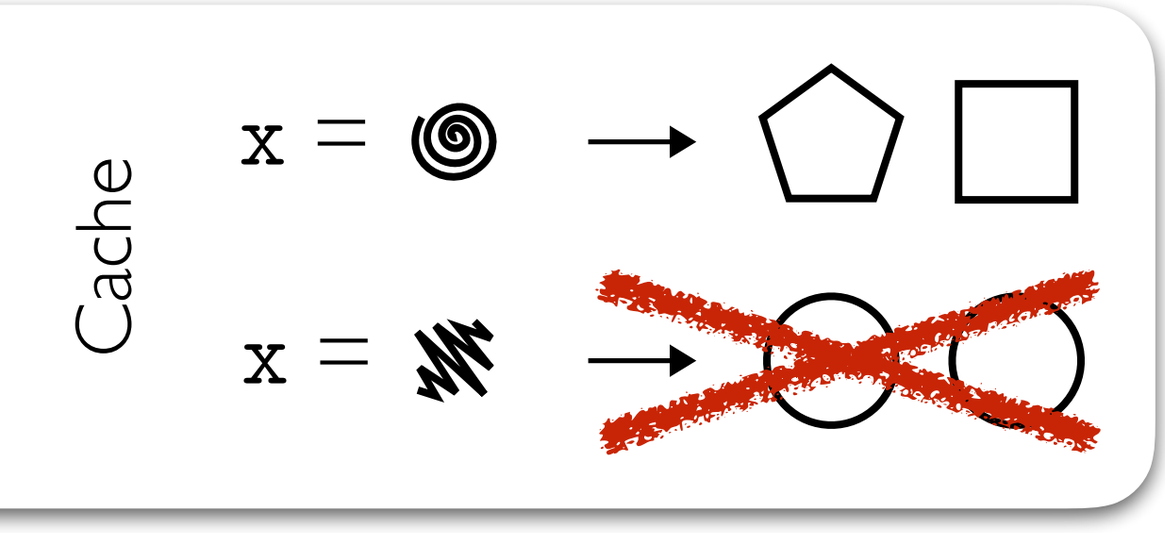
$$\Longrightarrow \mathtt{x} = \mathtt{z}$$

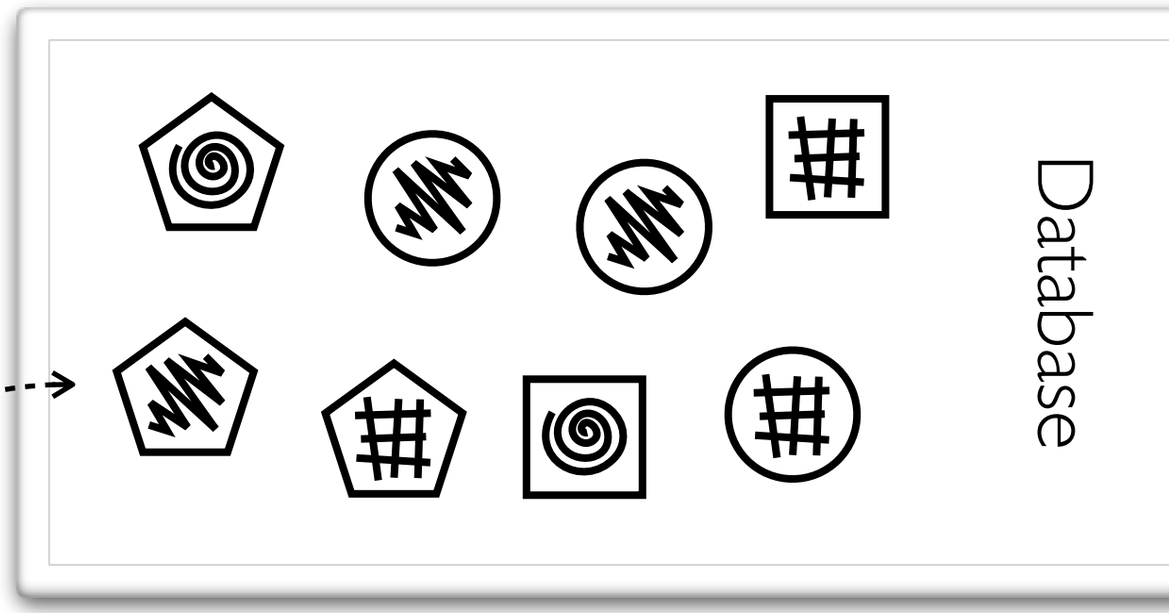
cache key

known during update

# Invalidation for INSERT

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




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

(`y`, `z`) =

**Invalidation formula:**

∃ (*shape*, *fill*).
*fill* = `x` ∧ *shape* = `y` ∧ *fill* = `z`

⟹ `x` = `z`

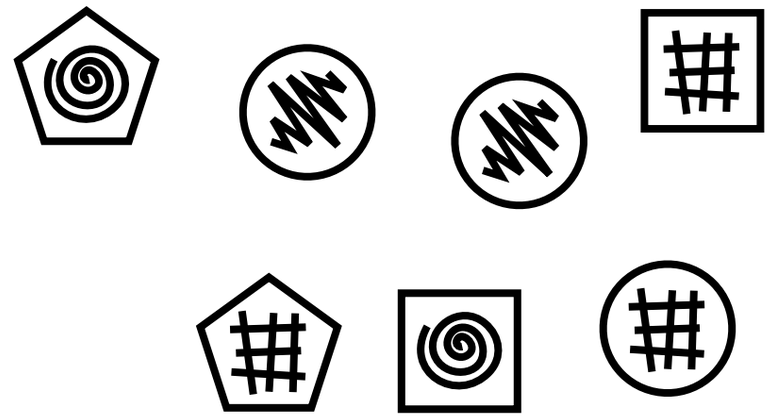
inval(`z`);

# Invalidation for UPDATE

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

Cache

Database



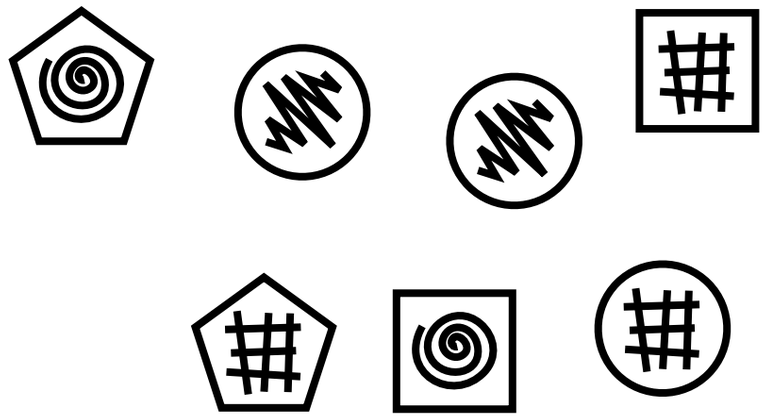
UPDATE *fill* = `y` WHERE *fill* = `z`

# Invalidation for UPDATE

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

Cache

`x` =

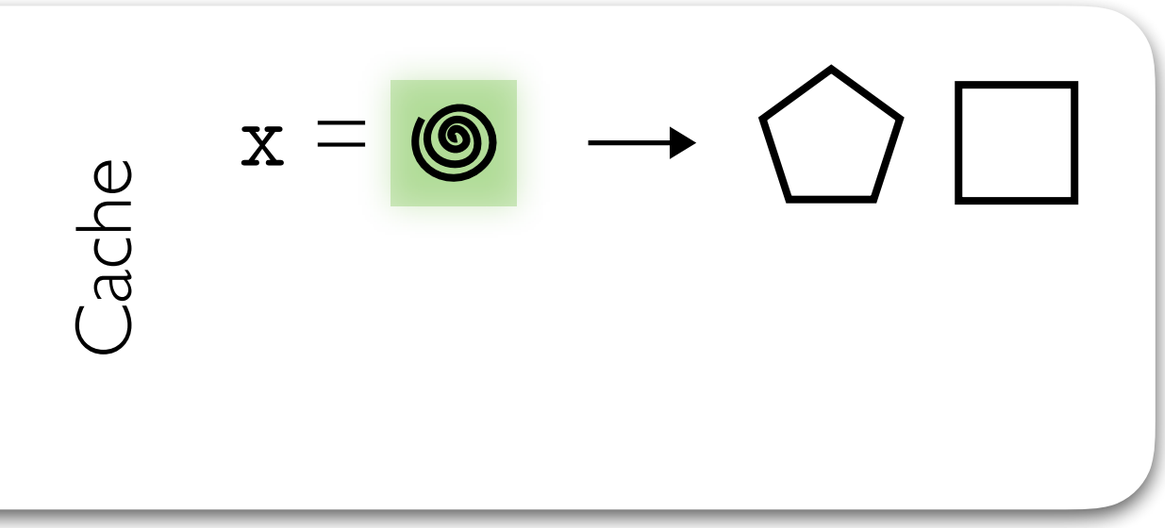


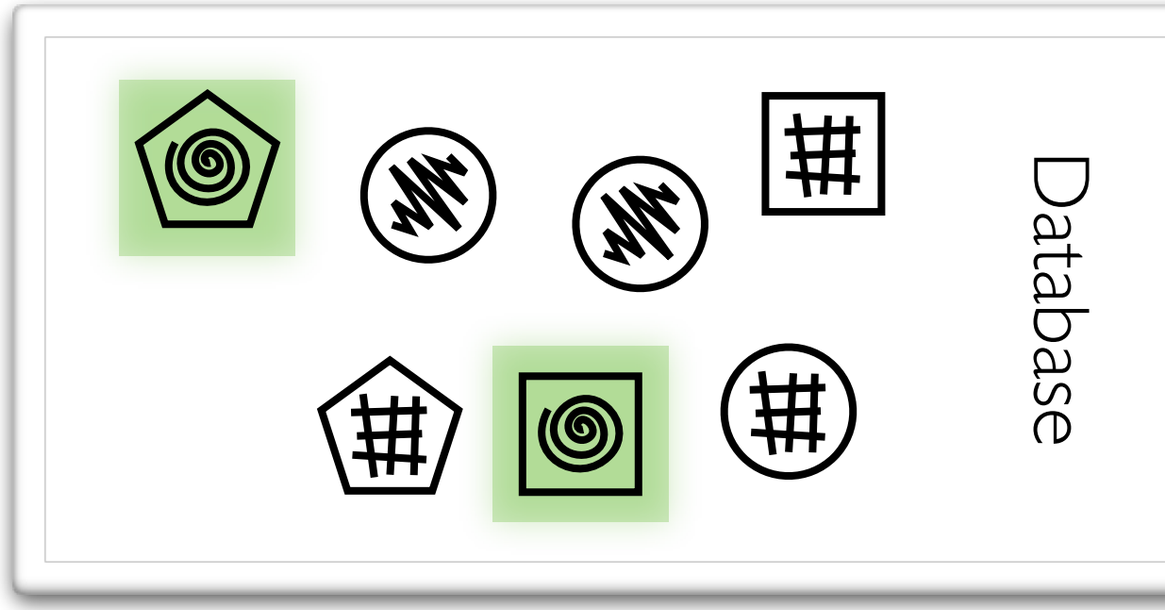
Database

UPDATE *fill* = `y` WHERE *fill* = `z`

# Invalidation for UPDATE

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

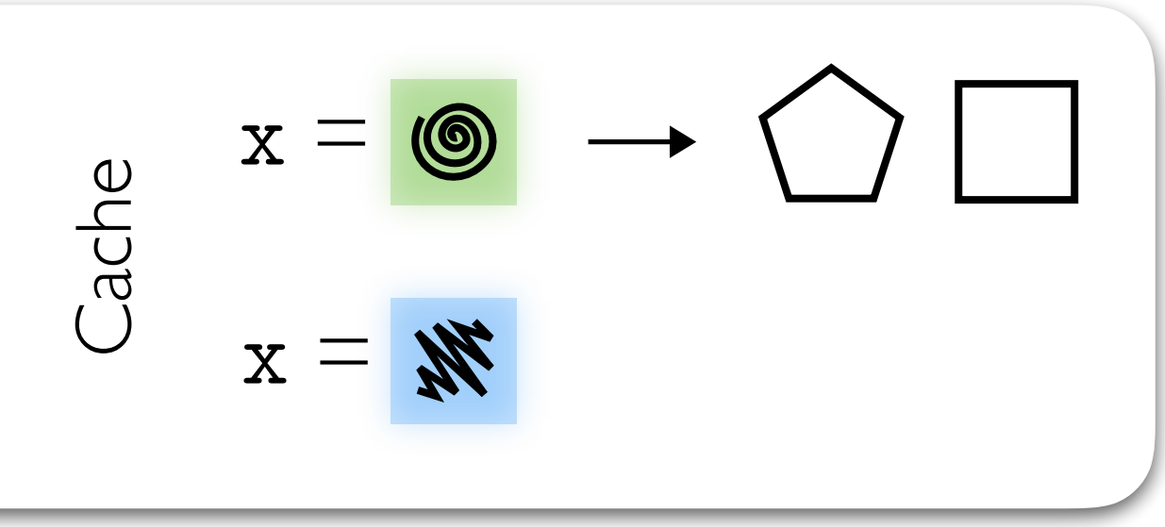


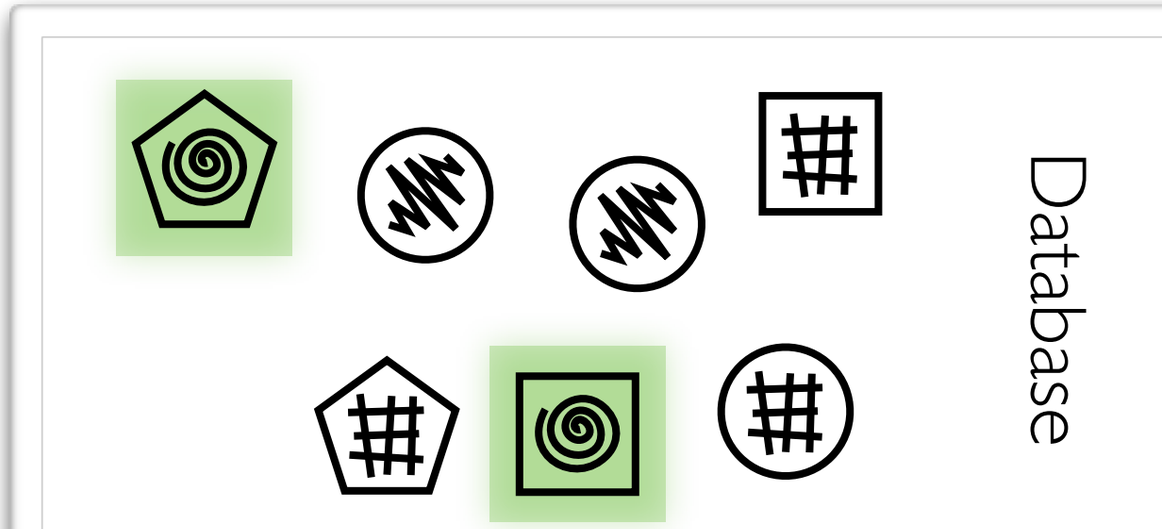




UPDATE *fill* = **y** WHERE *fill* = **z**

# Invalidation for UPDATE

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

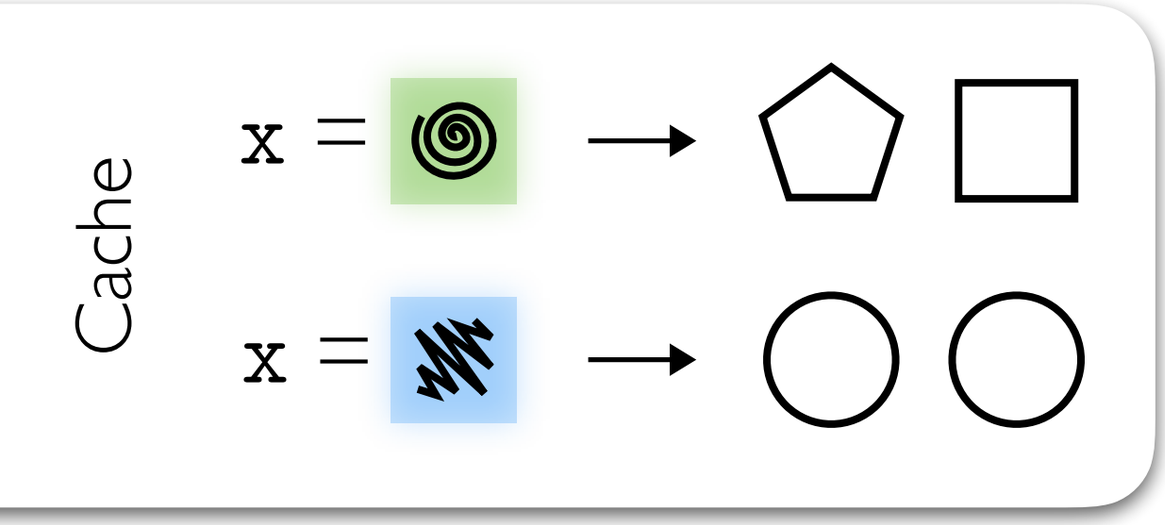


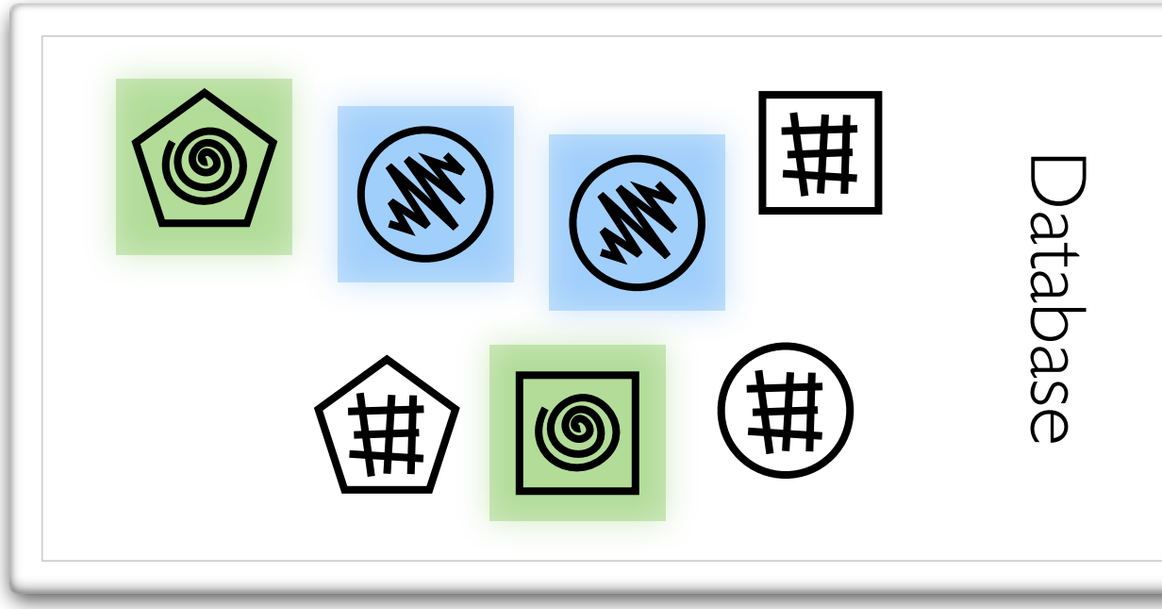




UPDATE *fill* = `y` WHERE *fill* = `z`

# Invalidation for UPDATE

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

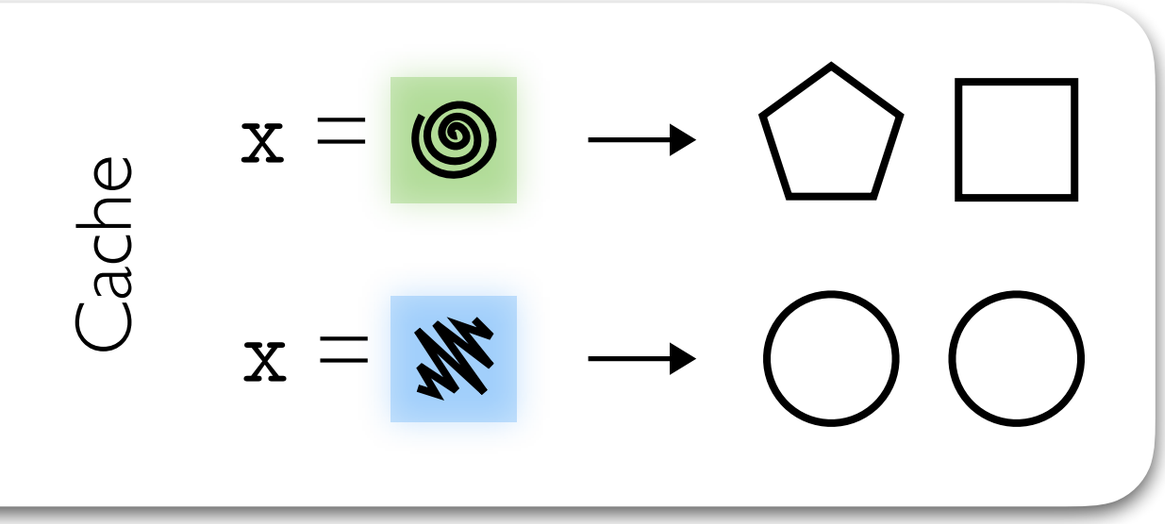


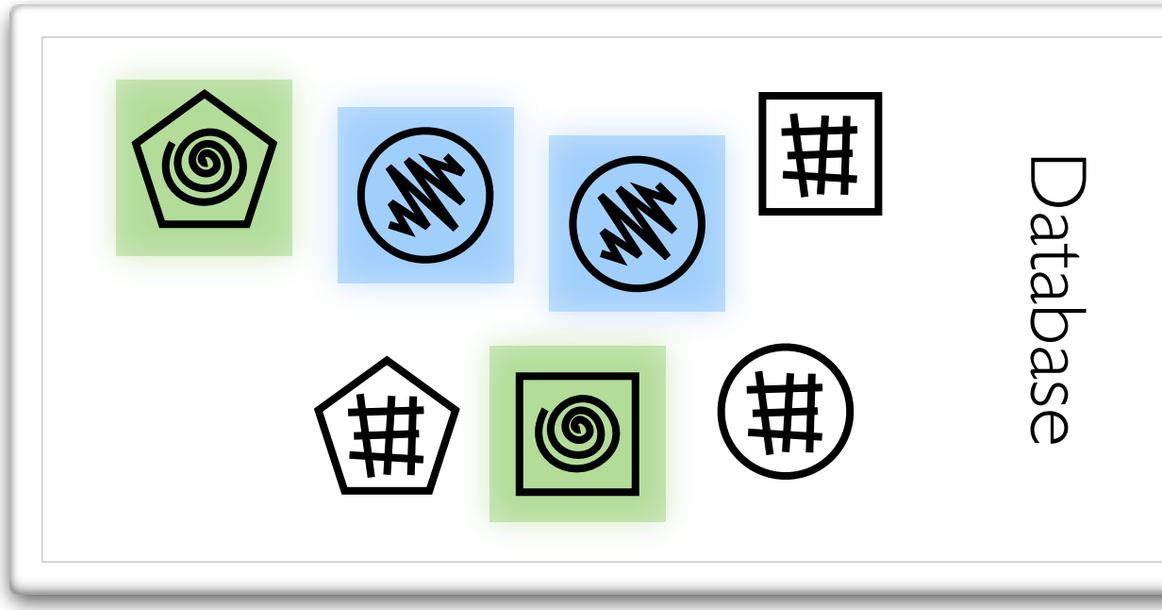




UPDATE *fill* = `y` WHERE *fill* = `z`

# Invalidation for UPDATE

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



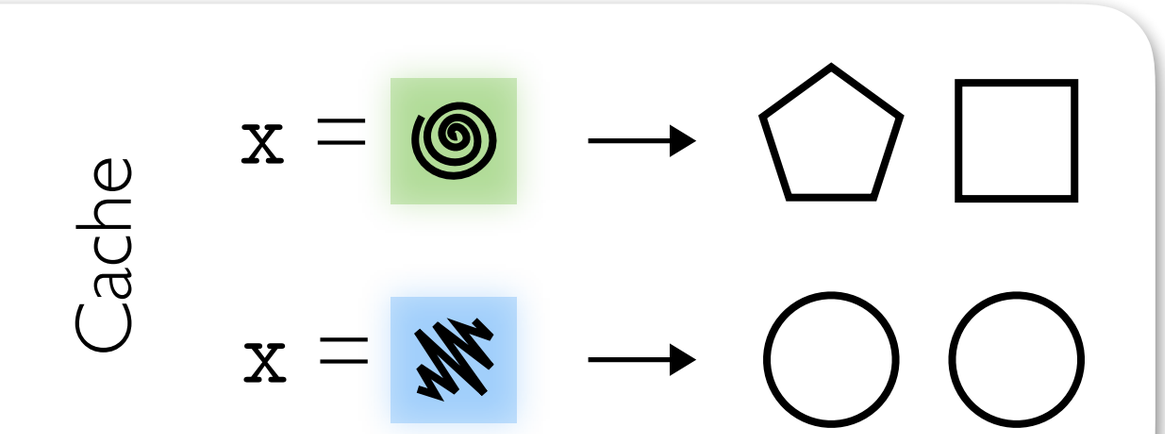



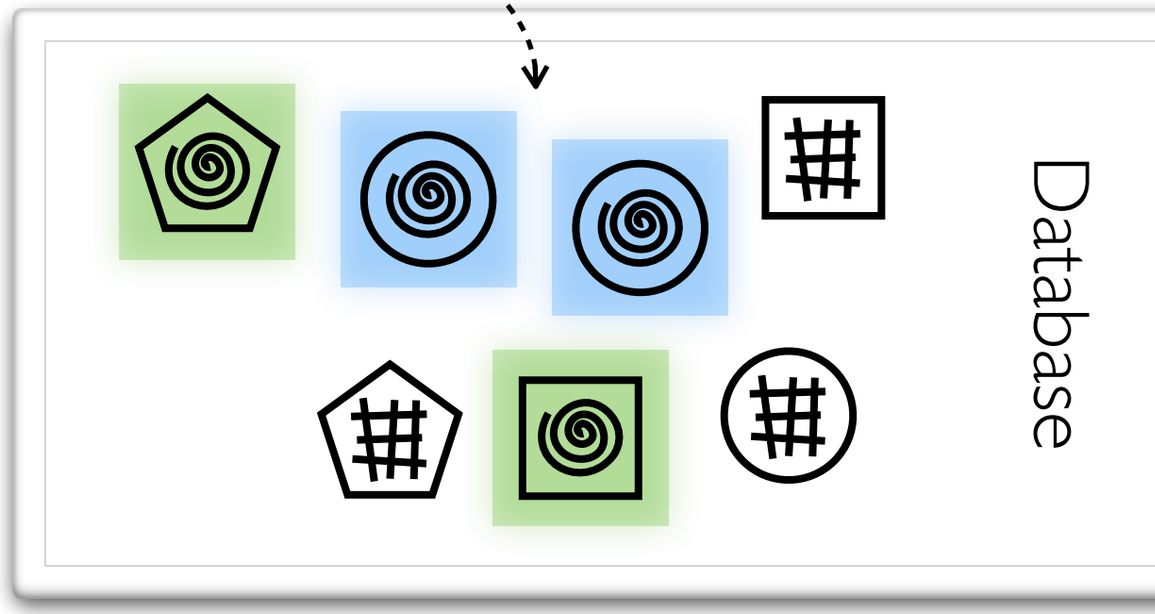

UPDATE *fill* = `y` WHERE *fill* = `z`

`y` = `z` =

# Invalidation for UPDATE

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



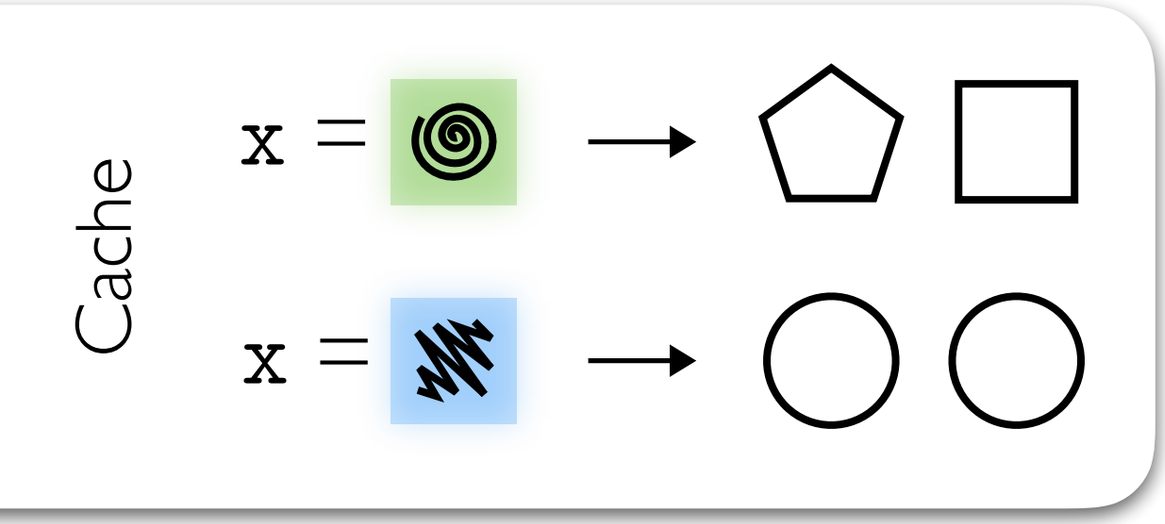



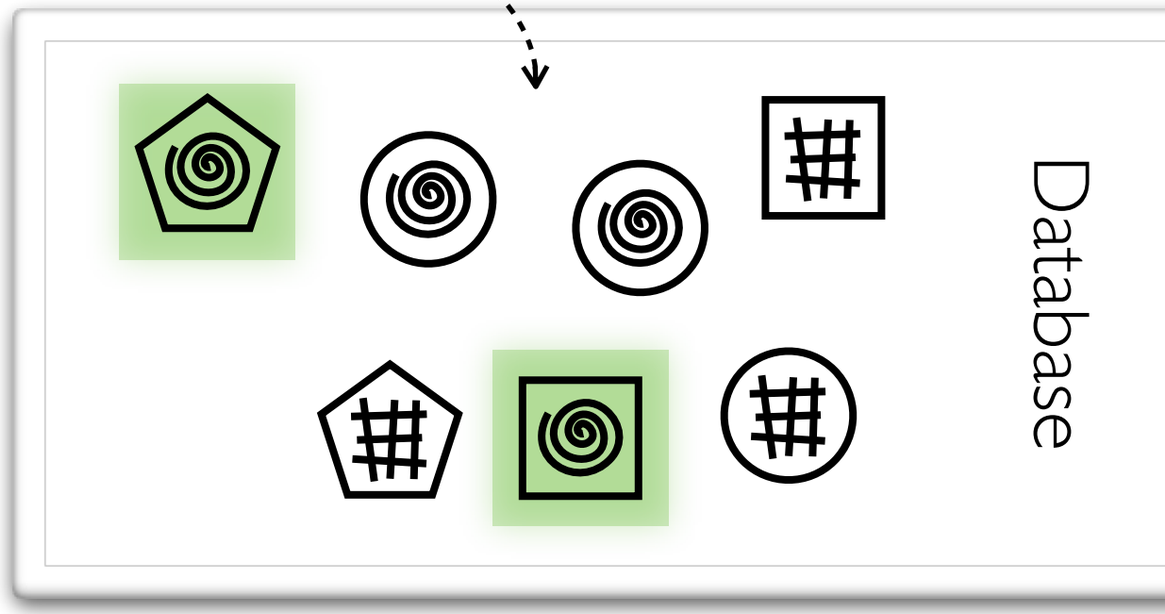

UPDATE *fill* = `y` WHERE *fill* = `z`

`y` = `z` =

# Invalidation for UPDATE

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



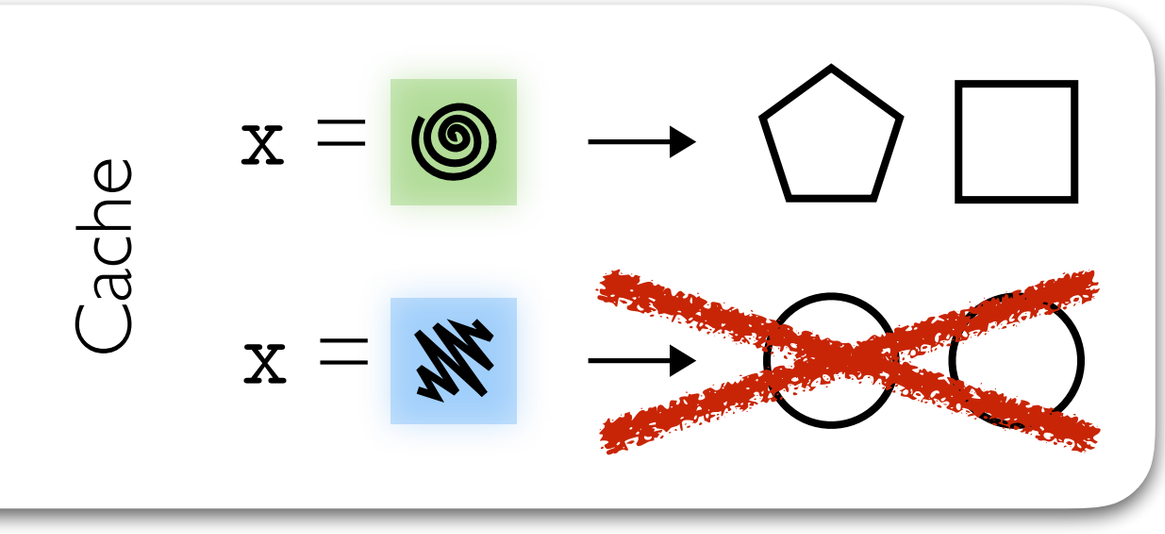



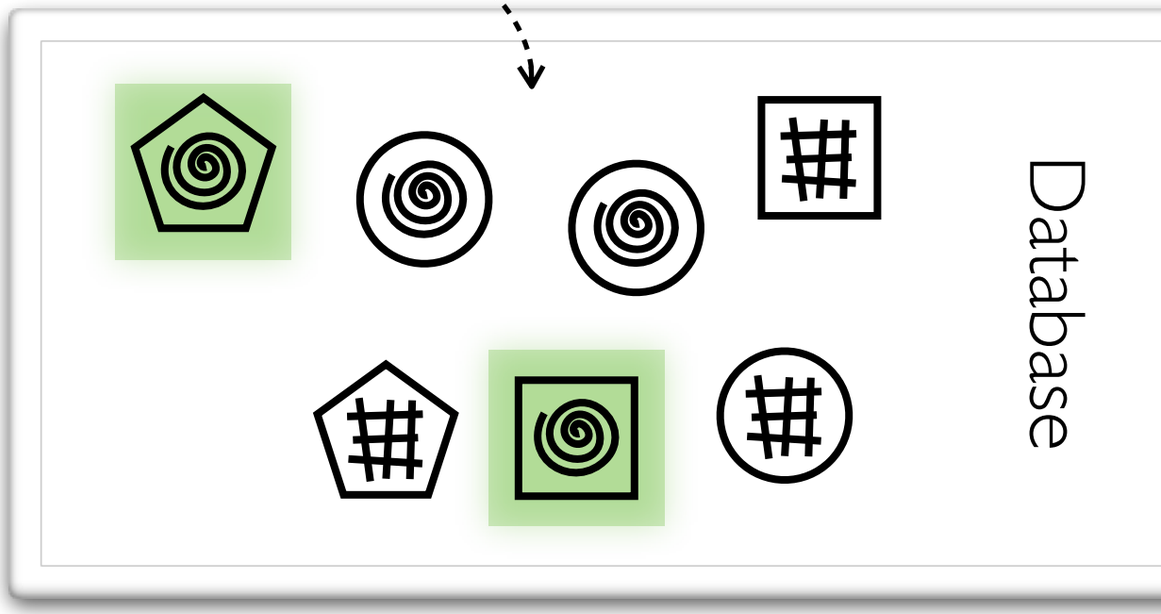

UPDATE *fill* = `y` WHERE *fill* = `z`

`y` = `z` =

# Invalidation for UPDATE

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



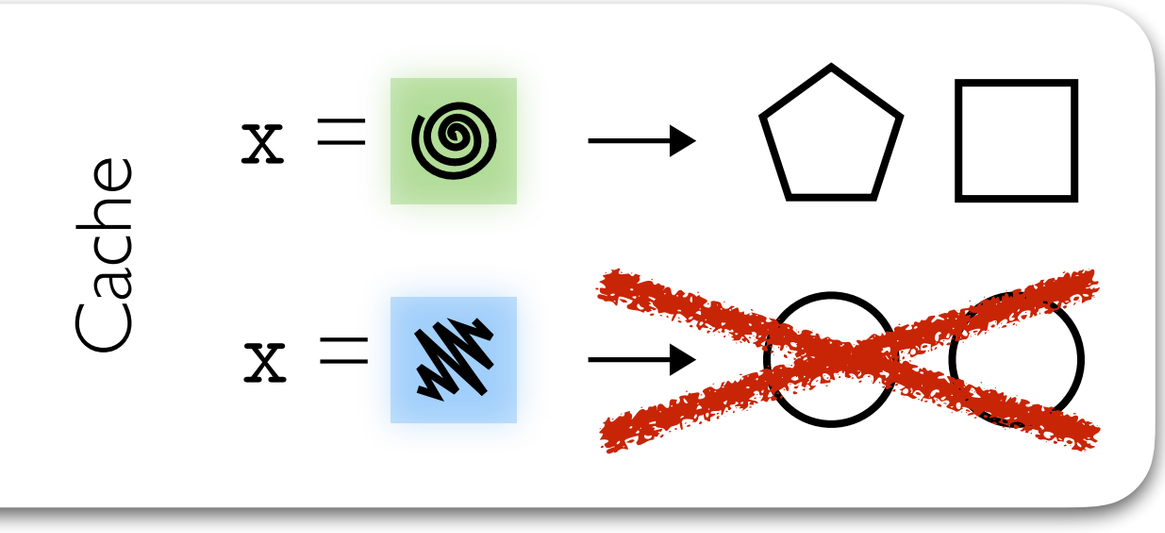



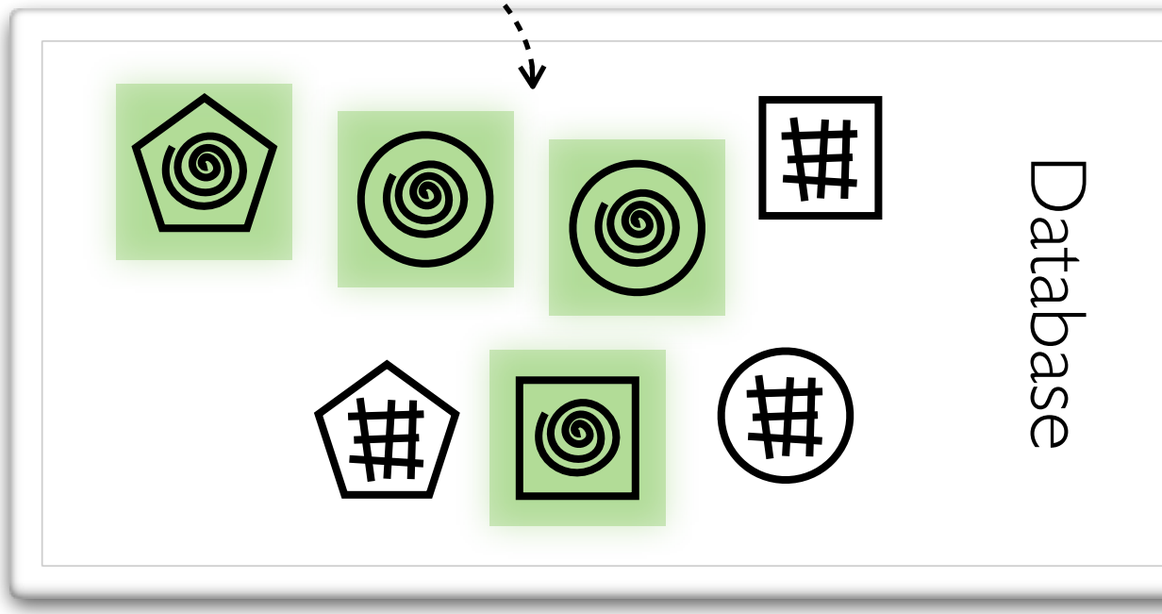

UPDATE *fill* = `y` WHERE *fill* = `z`

`y` = `z` =

# Invalidation for UPDATE

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



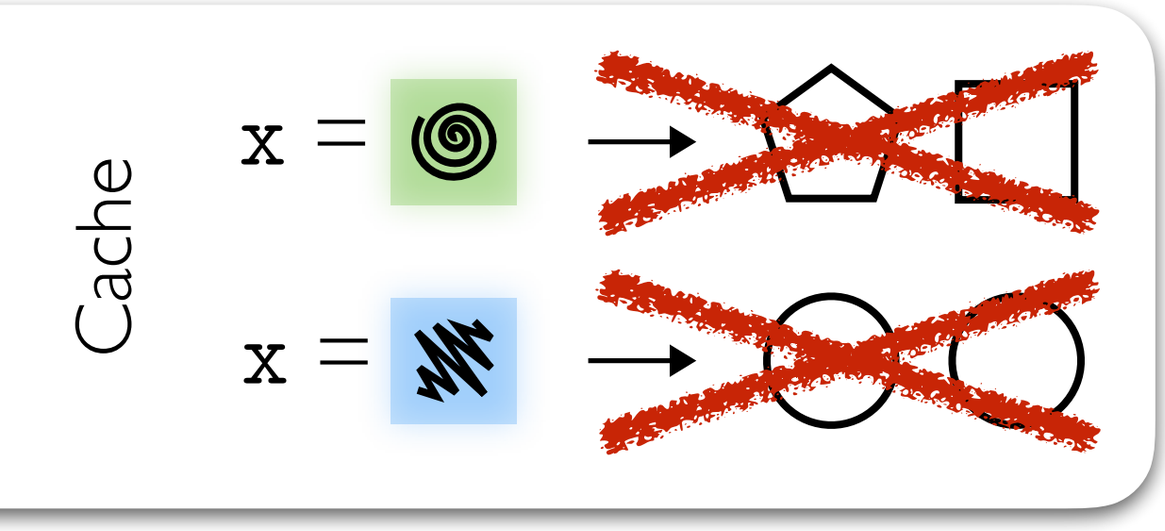



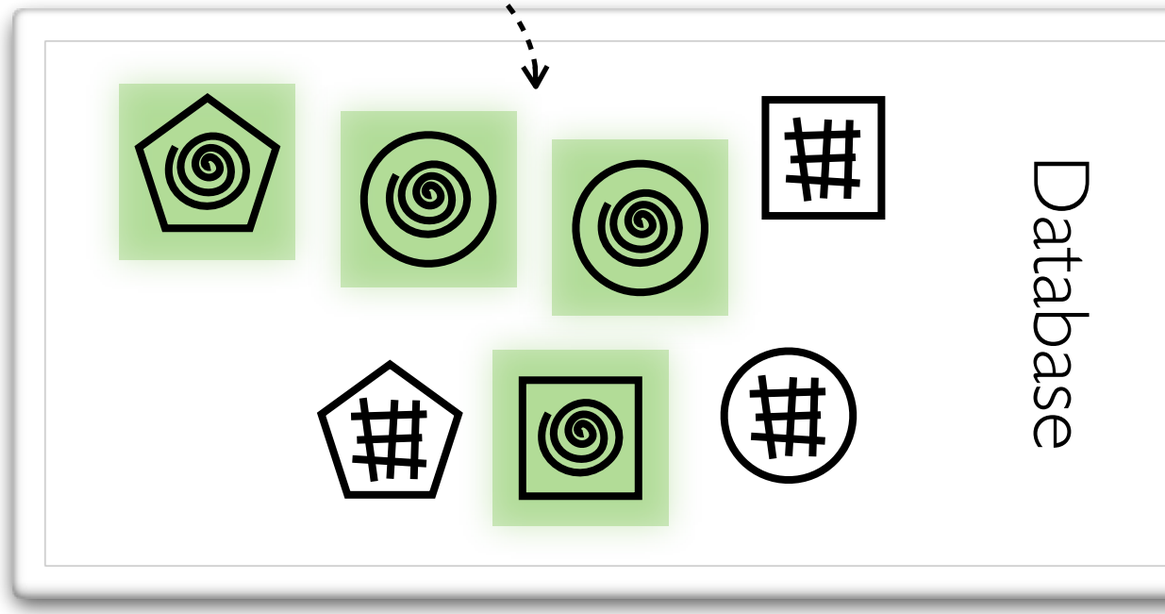

UPDATE *fill* = `y` WHERE *fill* = `z`

`y` = `z` =

# Invalidation for UPDATE

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



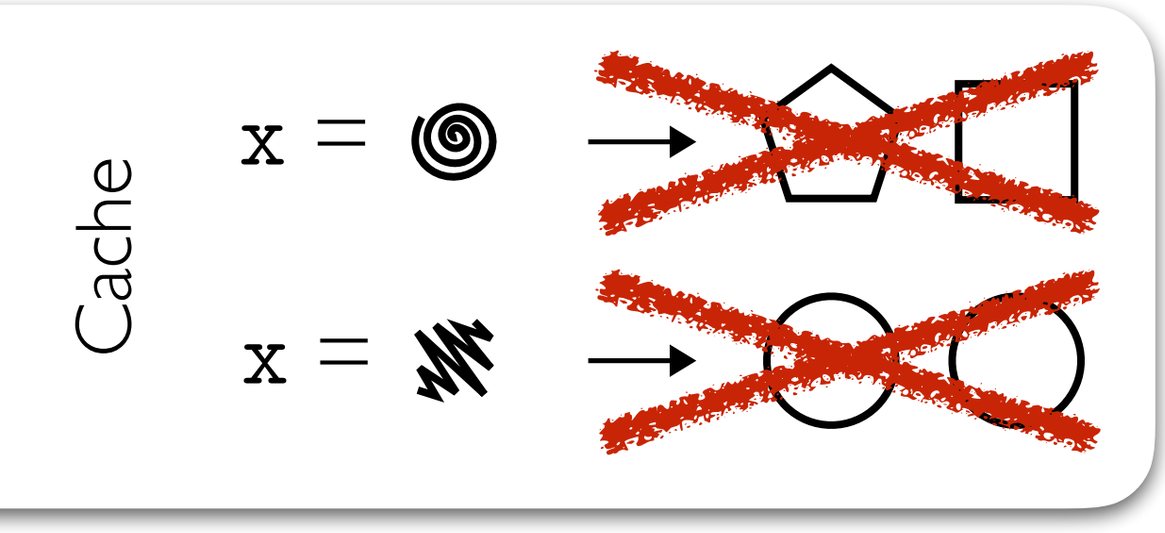



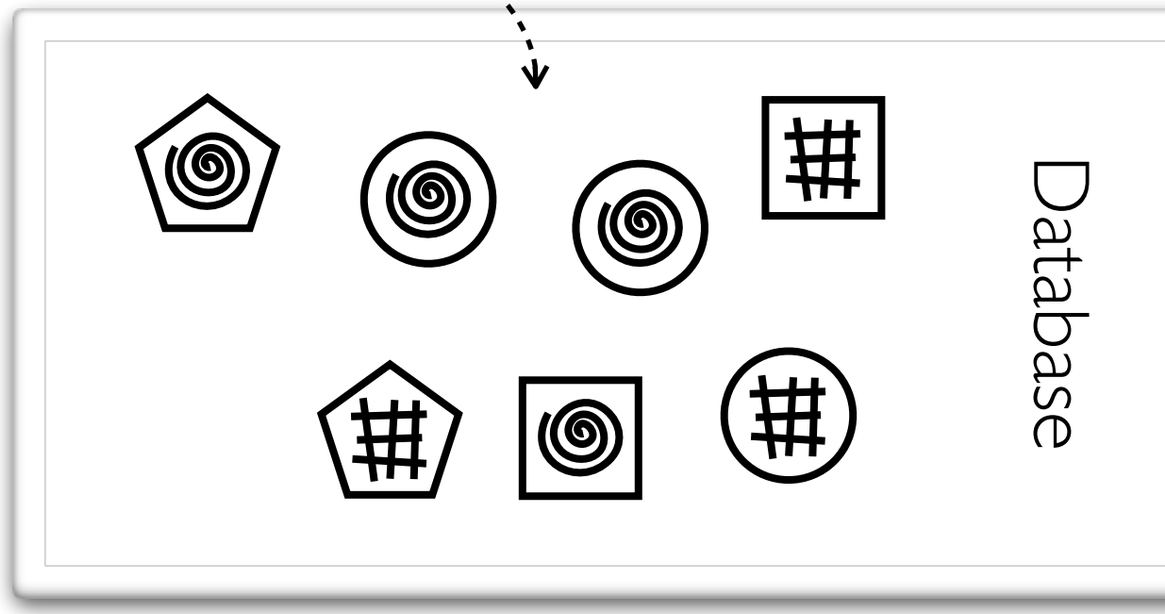

UPDATE *fill* = `y` WHERE *fill* = `z`

`y` = `z` =

# 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` =

Database

UPDATE *fill* = `y` WHERE *fill* = `z`

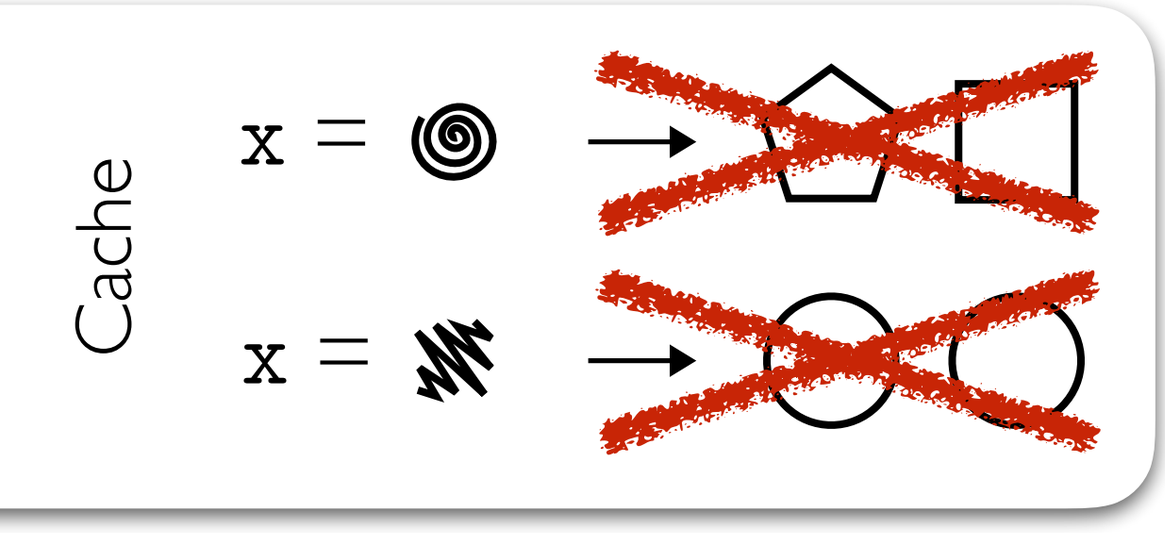
`y` = `z` =

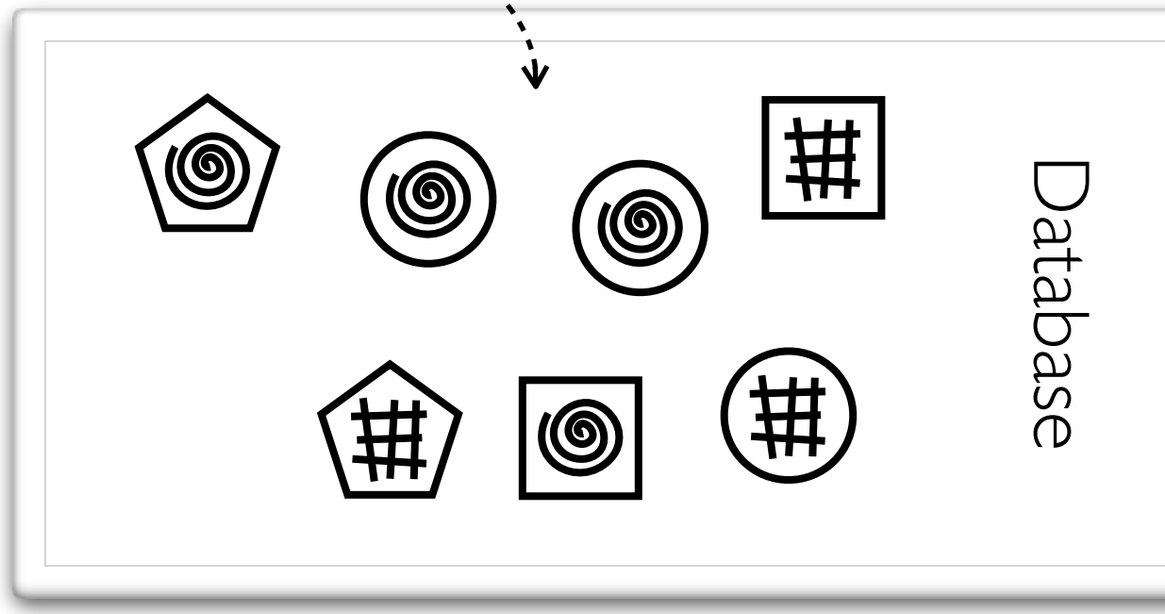
**Invalidation formula:**

$\exists$ (*shape*, *fill*), (*shape*', *fill*').
(*fill* = `x` $\vee$ *fill*' = `x`)
$\wedge$ (*fill*' = `y` $\wedge$ *fill* = `z`)

# Invalidation for UPDATE

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




UPDATE *fill* = `y` WHERE *fill* = `z`

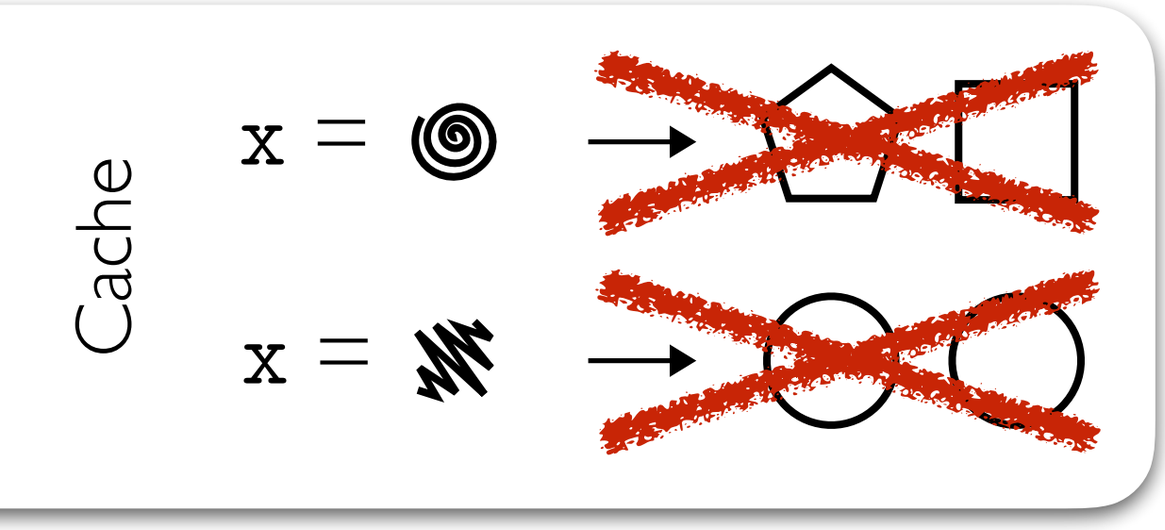
`y` = `z` =

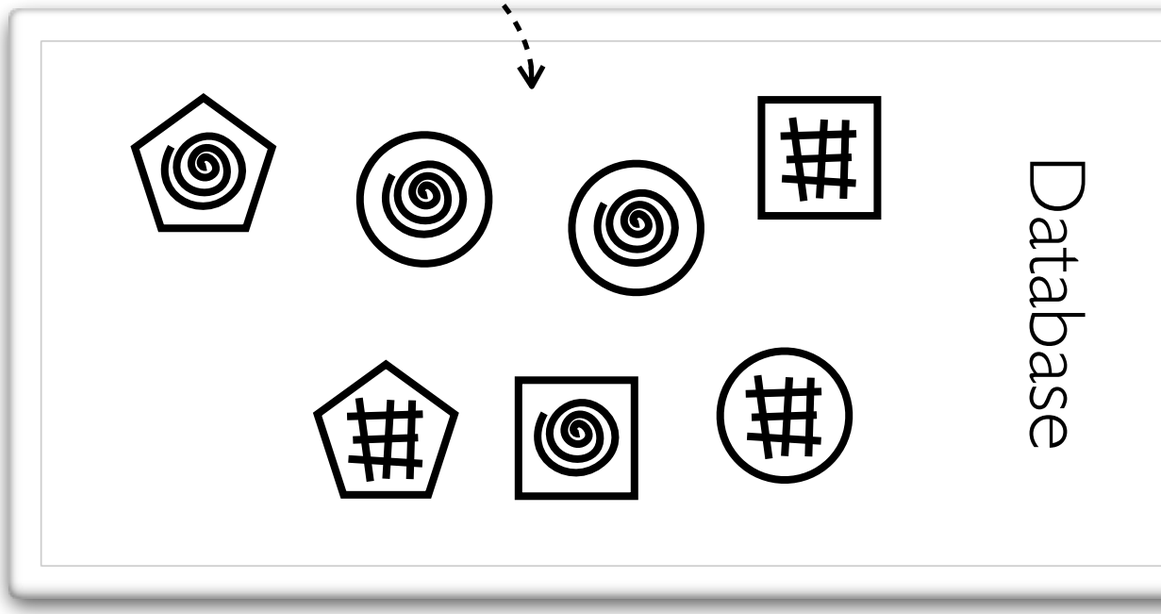
**Invalidation formula:**

$$\exists\ (shape, fill), (shape', fill').\ (fill = \mathtt{x} \vee fill' = \mathtt{x}) \wedge (fill' = \mathtt{y} \wedge fill = \mathtt{z}) \Longrightarrow \mathtt{x} = \mathtt{y} \vee \mathtt{x} = \mathtt{z}$$

# Invalidation for UPDATE

SELECT *shape* WHERE *fill* = x




UPDATE *fill* = y WHERE *fill* = z

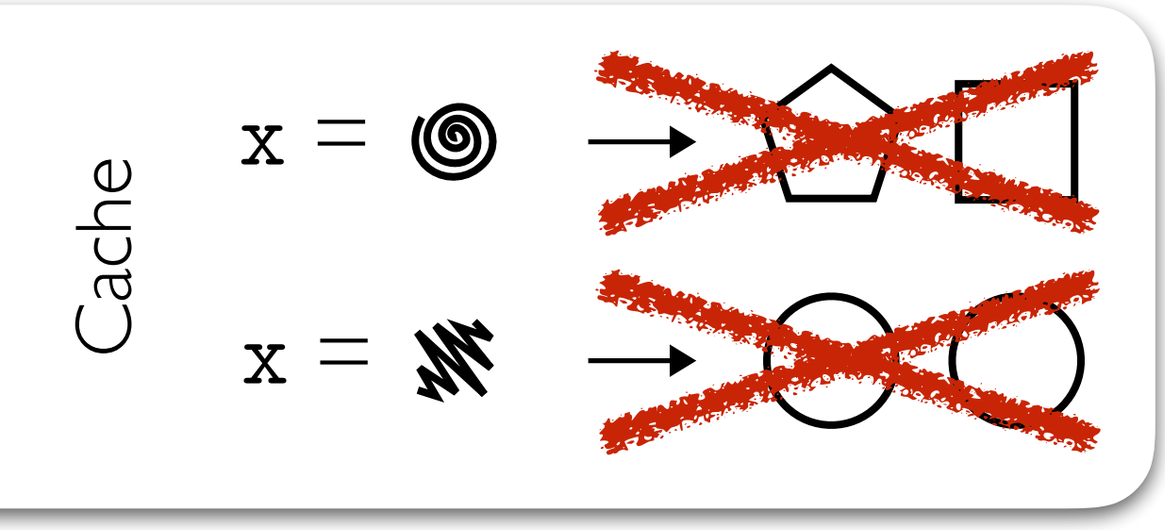
y = z =

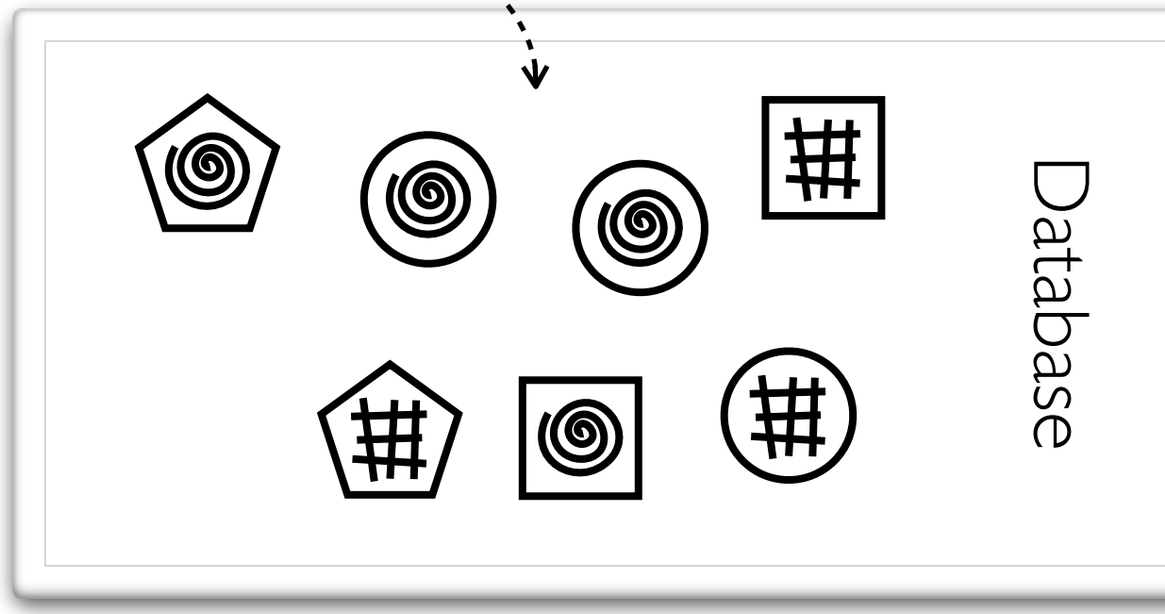
**Invalidation formula:**

$$\exists\ (shape, fill), (shape', fill').\ (fill = x \vee fill' = x) \wedge (fill' = y \wedge fill = z) \Longrightarrow x = y \vee x = z$$

# Invalidation for UPDATE

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




UPDATE *fill* = `y` WHERE *fill* = `z`

`y` = `z` =

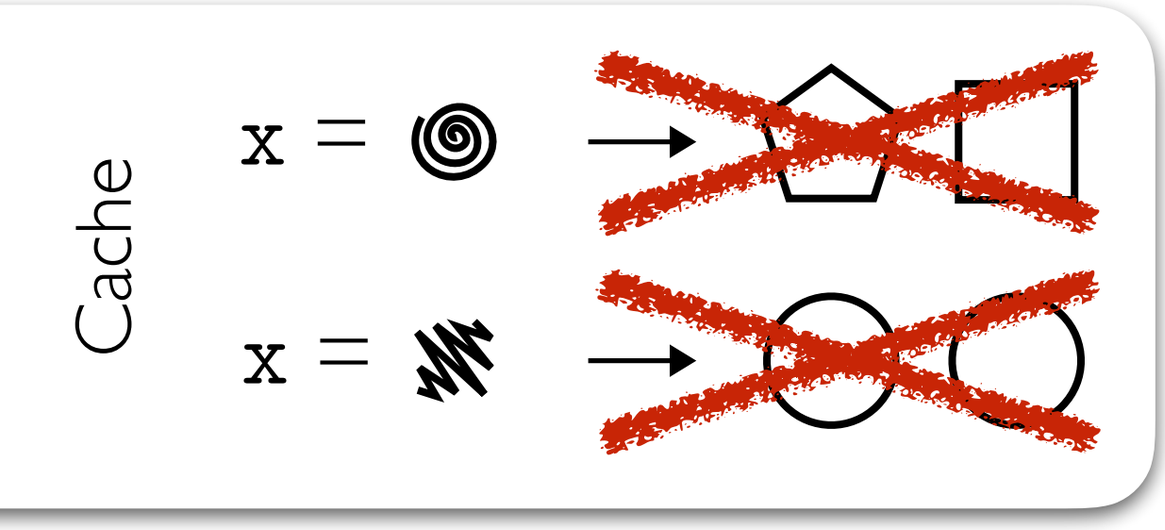
**Invalidation formula:**

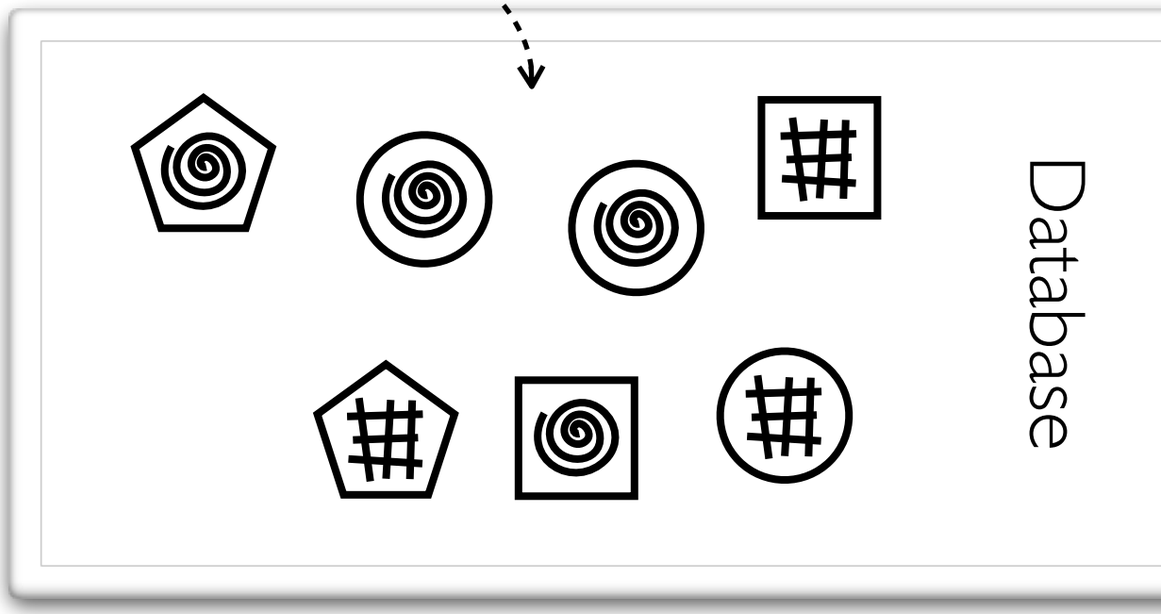
$$\exists\ (shape, fill), (shape', fill').\ (fill = \mathtt{x} \vee fill' = \mathtt{x}) \wedge (fill' = \mathtt{y} \wedge fill = \mathtt{z})$$

$$\Longrightarrow \mathtt{x} = \mathtt{y} \vee \mathtt{x} = \mathtt{z}$$

# Invalidation for UPDATE

SELECT *shape* WHERE *fill* = x




UPDATE *fill* = y WHERE *fill* = z

y = z =

**Invalidation formula:**

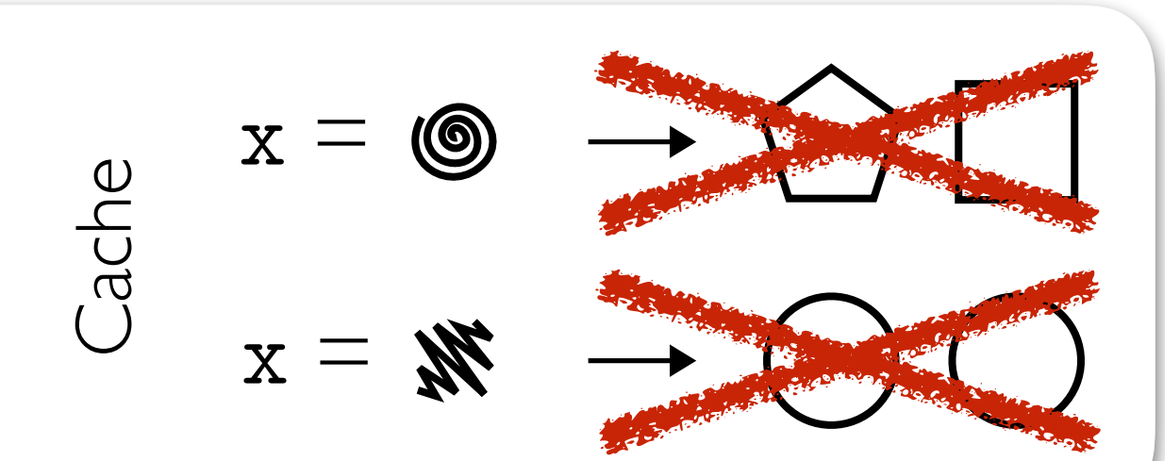
$$\exists\ (shape, fill), (shape', fill').\ (fill = x \vee fill' = x) \wedge (fill' = y \wedge fill = z)$$

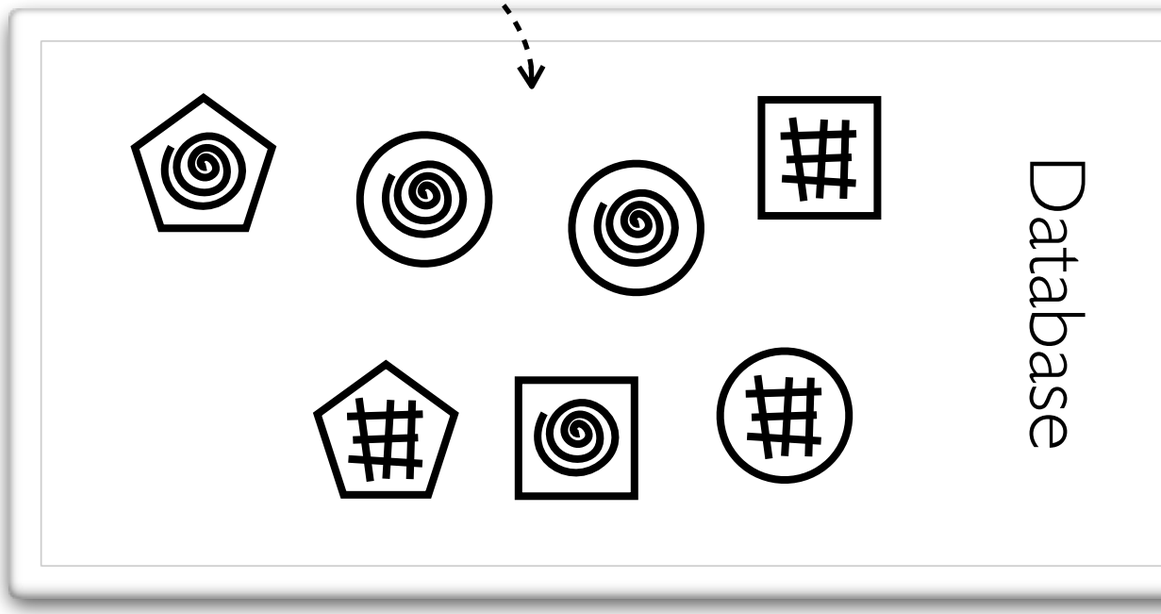
$$\Longrightarrow x = y \vee x = z$$

inval(y); inval(z);

# Invalidation for UPDATE

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




UPDATE *fill* = `y` WHERE *fill* = `z`

(*fill* = `x` ∧ *fill*' ≠ `x`) ∨ (*fill* ≠ `x` ∧ *fill*' = `x`)
∨ (*fill* = `x` ∧ *fill*' = `x` ∧ *shape* ≠ *shape*')

**Invalidation formula:**

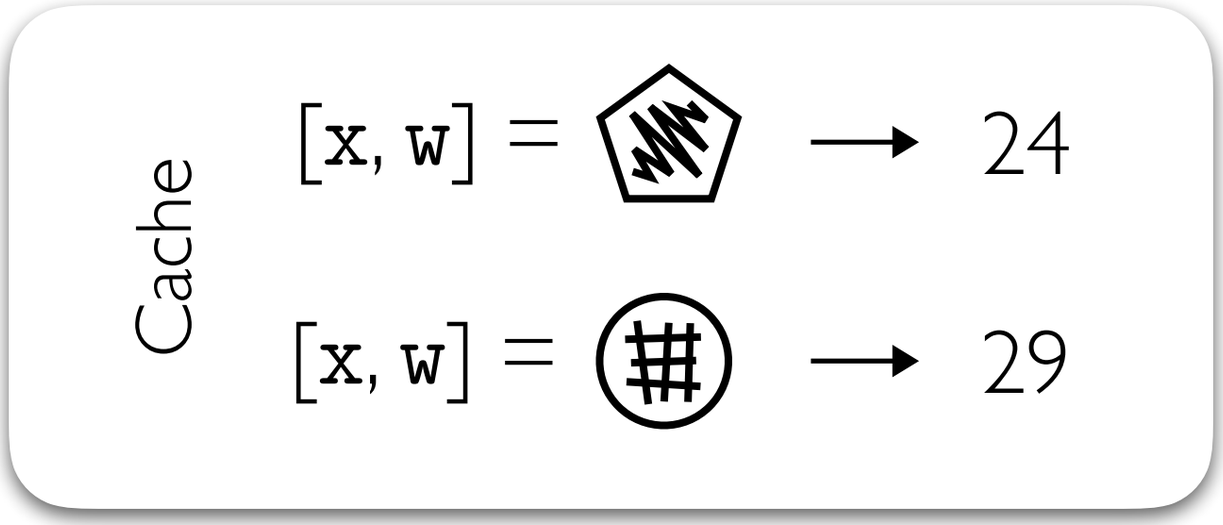
∃ (*shape*, *fill*), (*shape*', *fill*').
(*fill* = `x` ∨ *fill*' = `x`)
∧ (*fill*' = `y` ∧ *fill* = `z`)

⟹ `x` = `y` ∨ `x` = `z`

inval(`y`); inval(`z`);

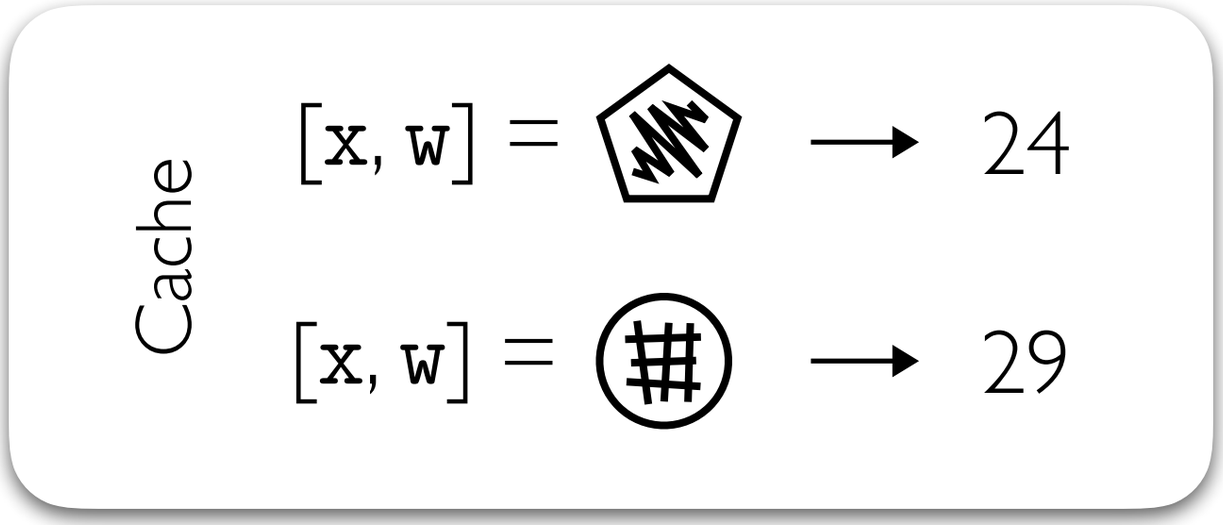
# Compound Cache Keys

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



# Compound Cache Keys

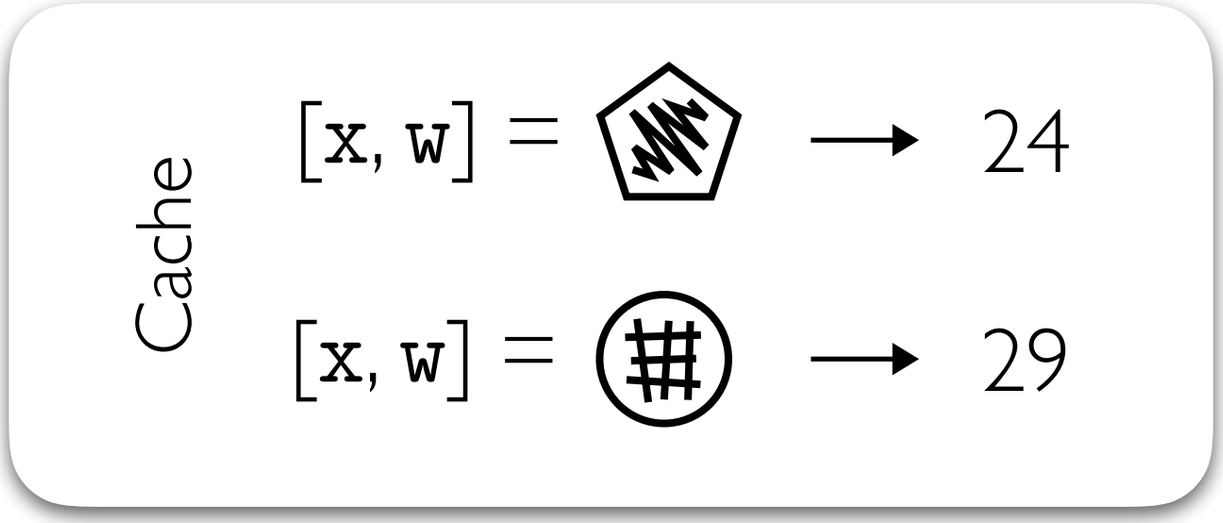
SELECT COUNT(*) WHERE *fill* = `x` ∧ *shape* = `w`




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

# Compound Cache Keys

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




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

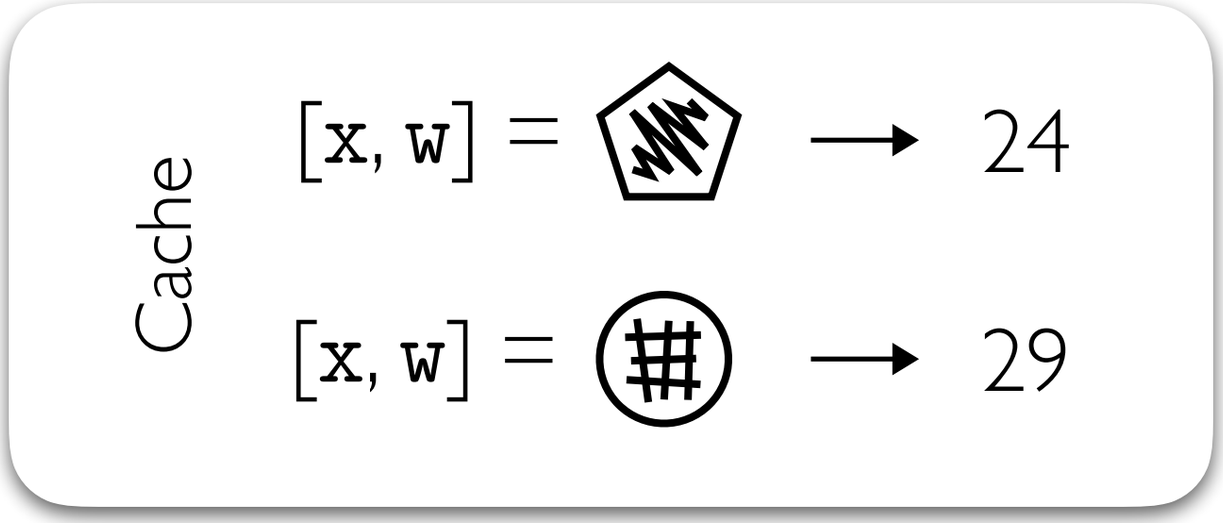
UPDATE *fill* = `y` WHERE *fill* = `z`

$\Longrightarrow$ `x` = `z` ∧ `w` = `y`

inval([`z`, `y`]);

# Compound Cache Keys

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




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

$\Longrightarrow$ `x` = `z` ∧ `w` = `y`

inval([`z`, `y`]);

UPDATE *fill* = `y` WHERE *fill* = `z`

$\Longrightarrow$ `x` = `y` ∨ `x` = `z`

inval([`y`, *]); inval([`z`, *]);

# Cache Data Structure

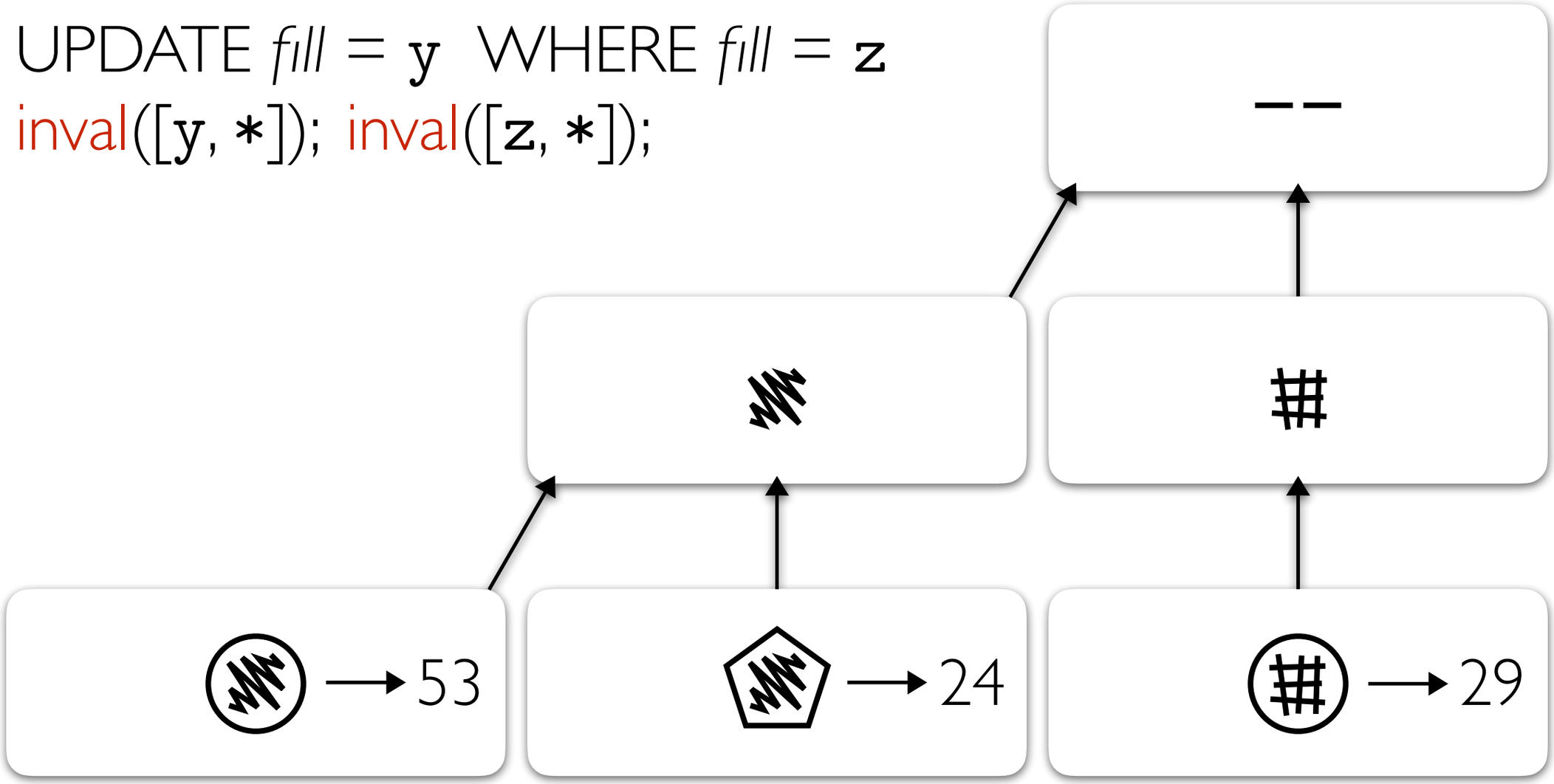
SELECT COUNT(*) WHERE *fill* = `x` ∧ *shape* = `w`

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

# Cache Data Structure

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

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



# Cache Data Structure

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

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

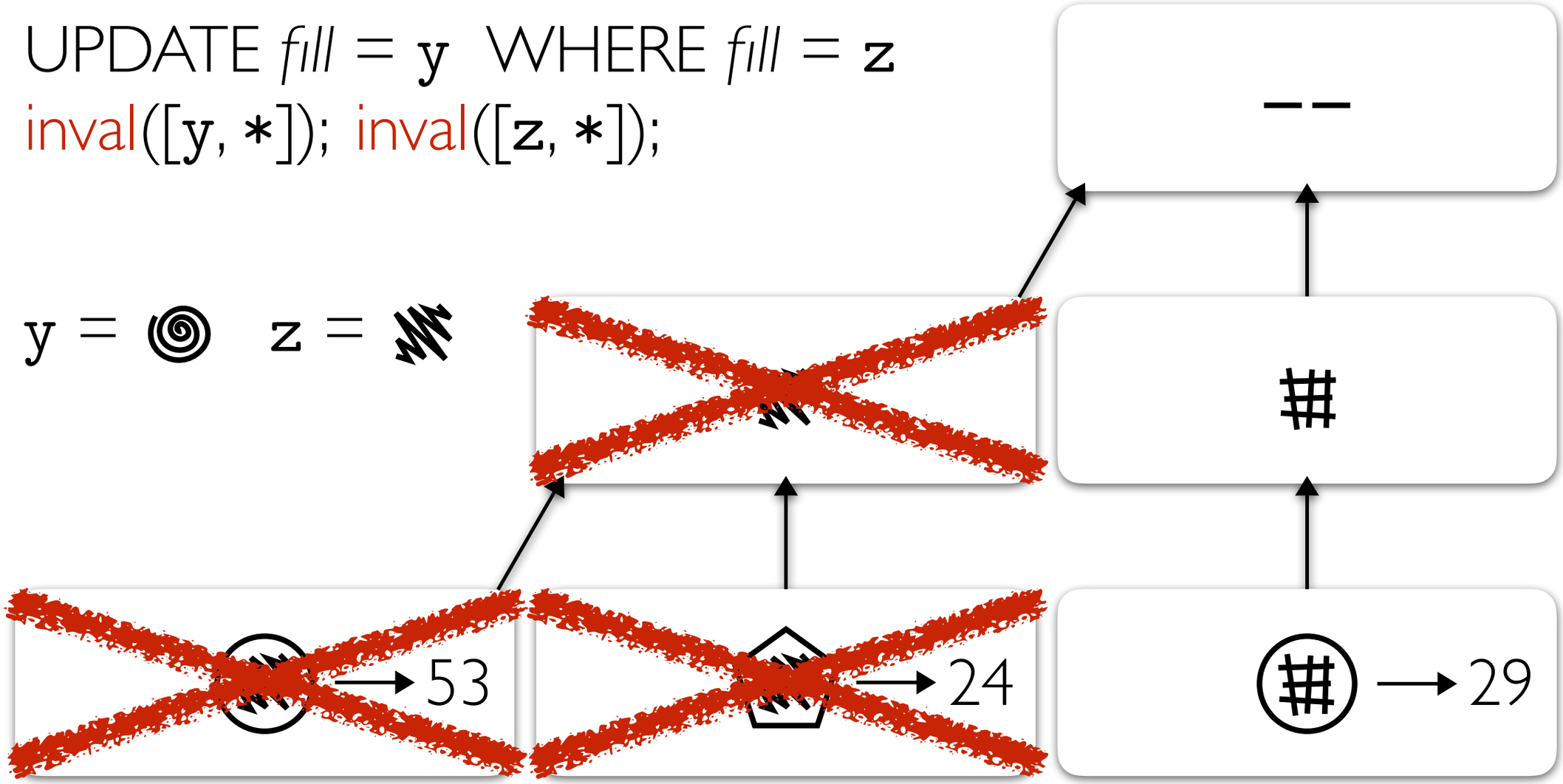
`y` =  `z` = 

# Cache Data Structure

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

UPDATE *fill* = `y` WHERE *fill* = `z`

inval([`y`, *]); inval([`z`, *]);



# Cache Data Structure

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

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

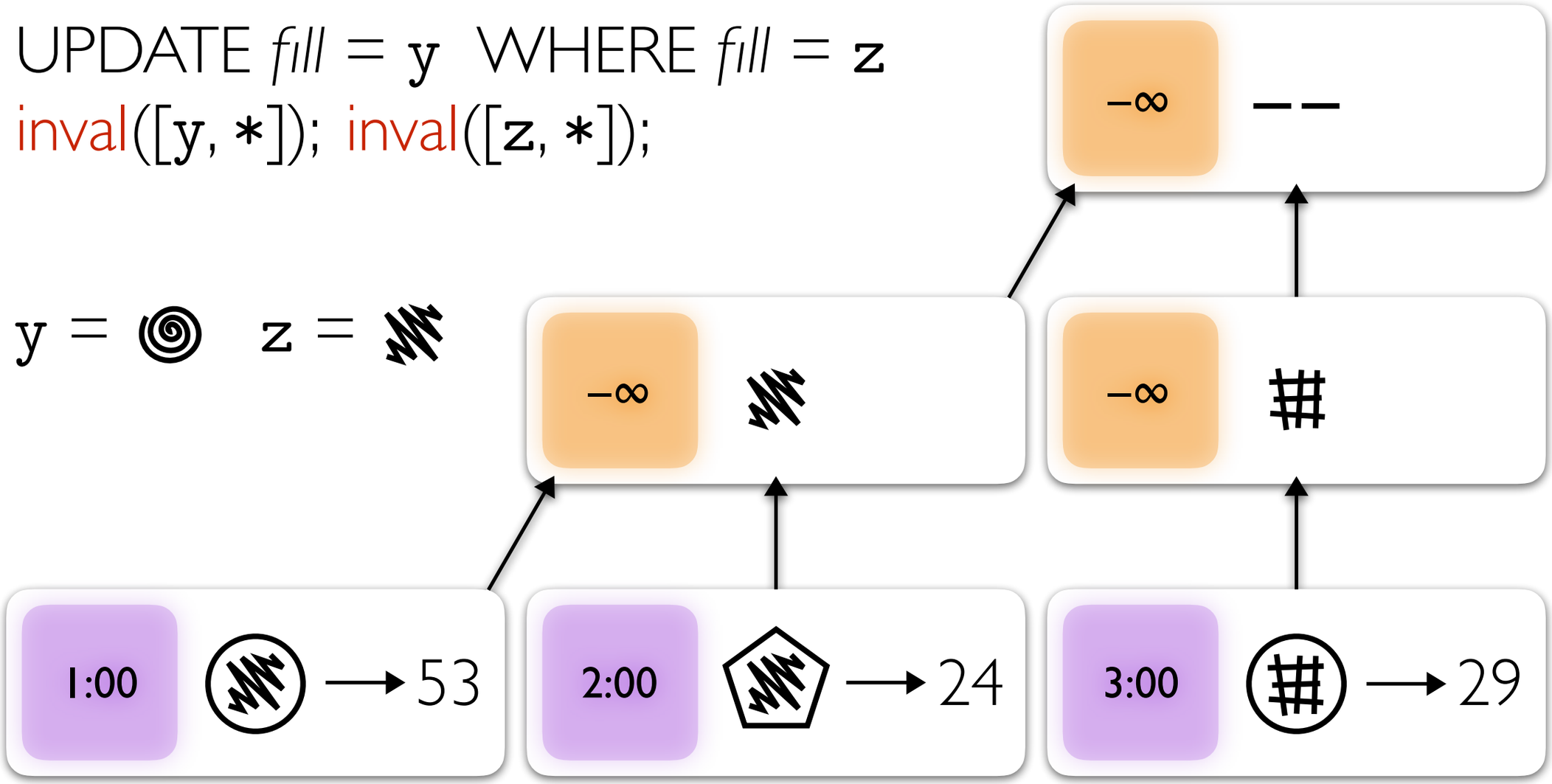
`y` =  `z` = 

# Cache Data Structure

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

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

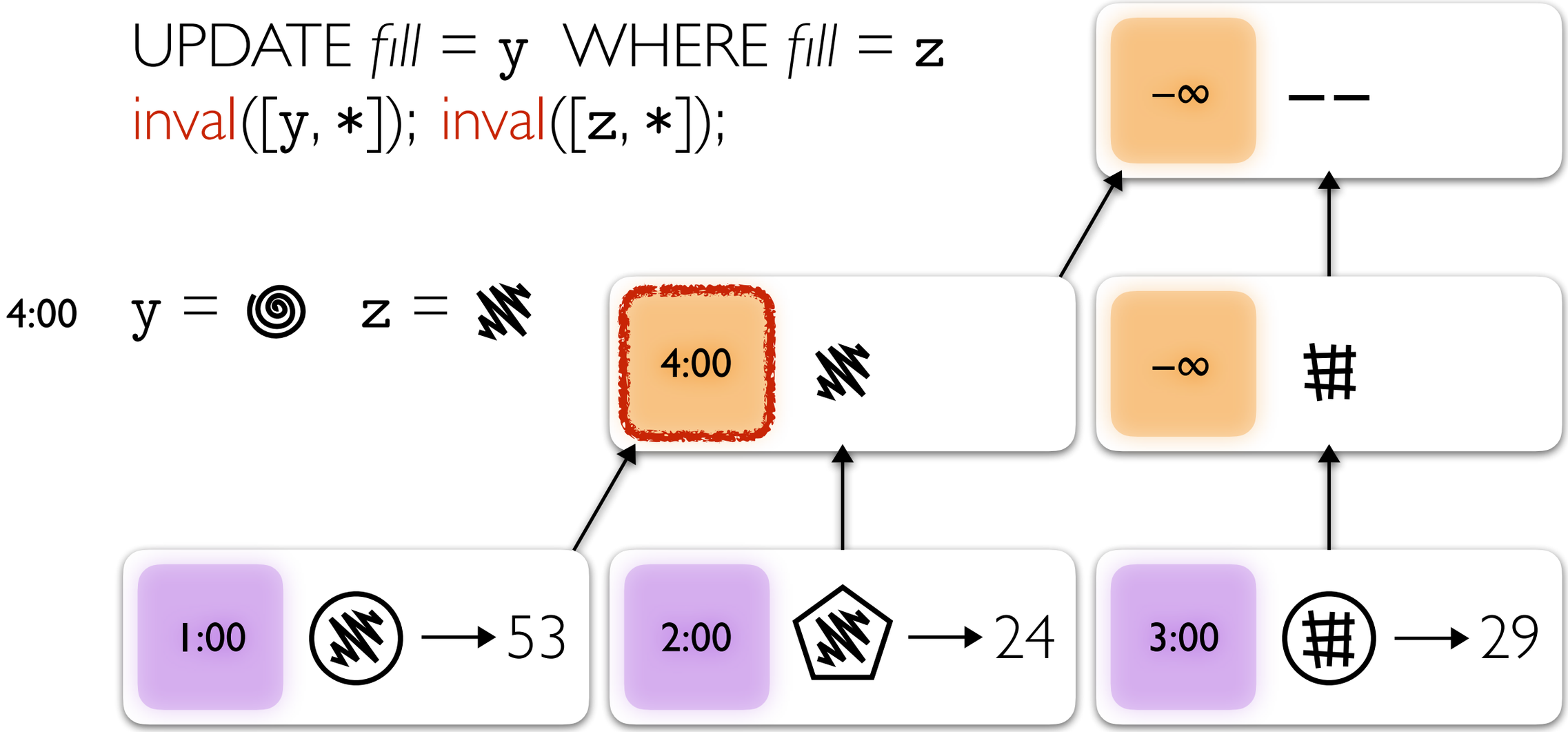
4:00 `y` = `z` =



# Cache Data Structure

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

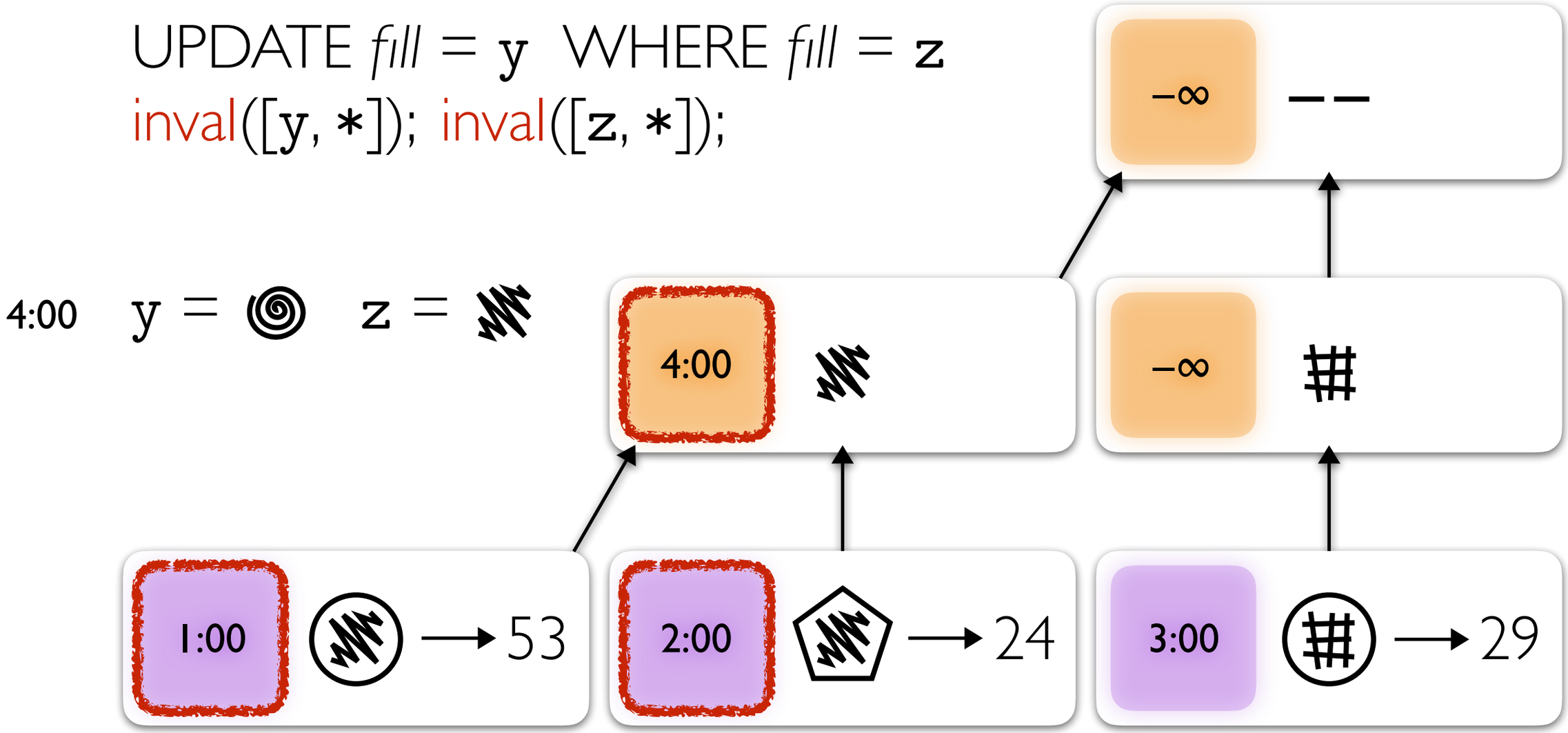
UPDATE *fill* = `y` WHERE *fill* = `z`
inval([`y`, *]); inval([`z`, *]);



# Cache Data Structure

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

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



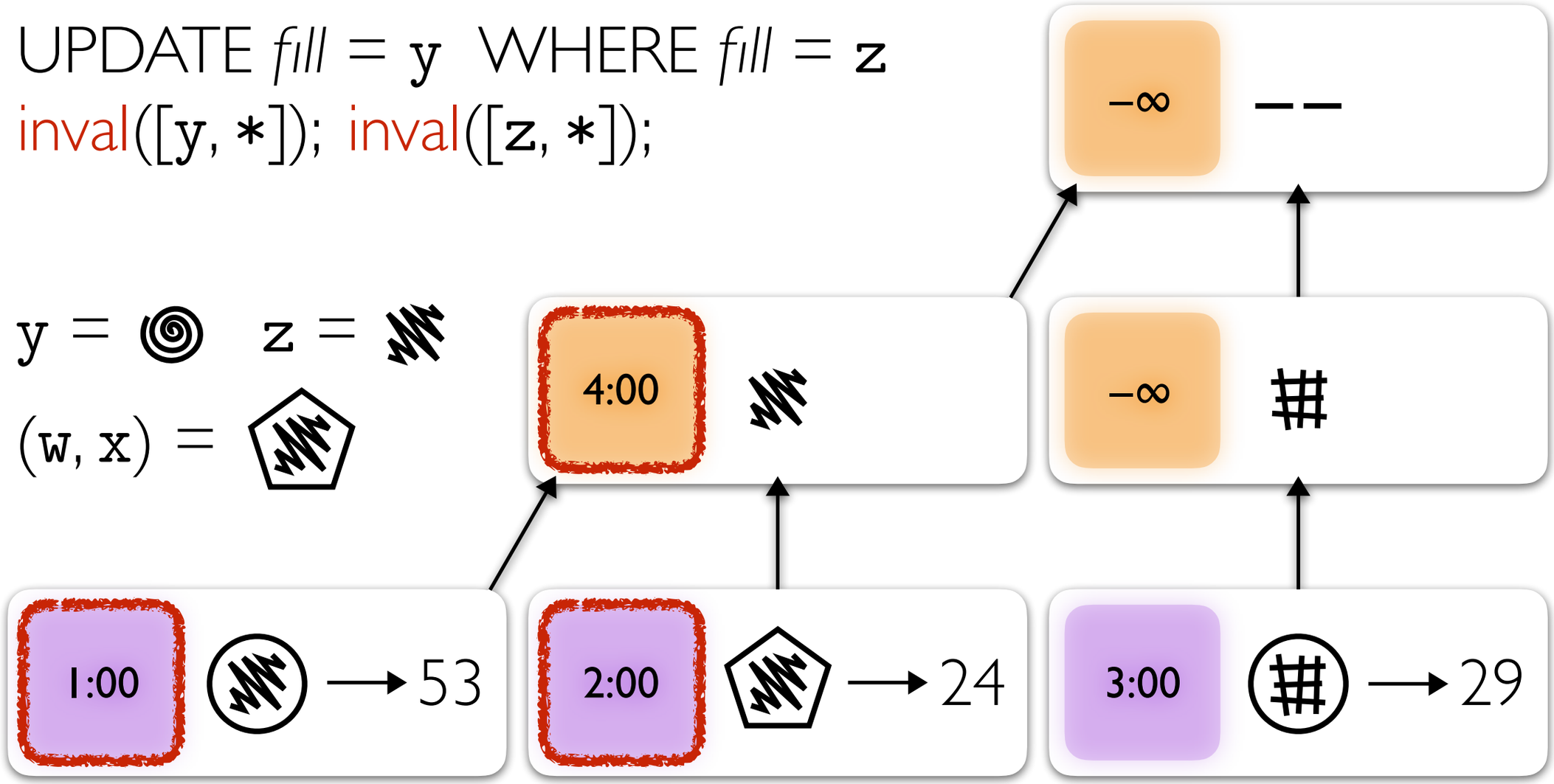
# Cache Data Structure

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

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

4:00 `y` = ◎ `z` = ≋

5:00 (`w`, `x`) = ⬠≋



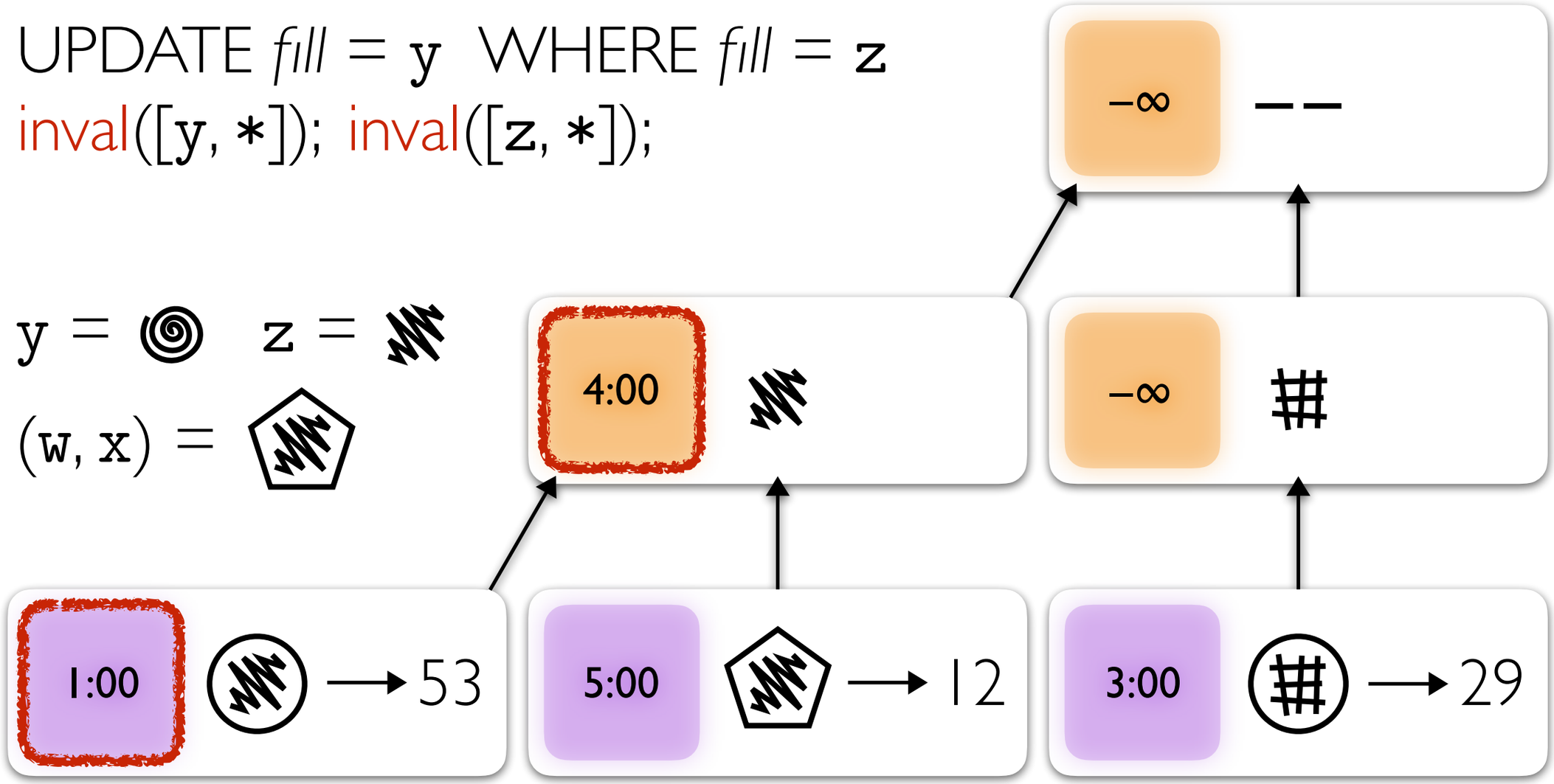
# Cache Data Structure

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`) = ⬠



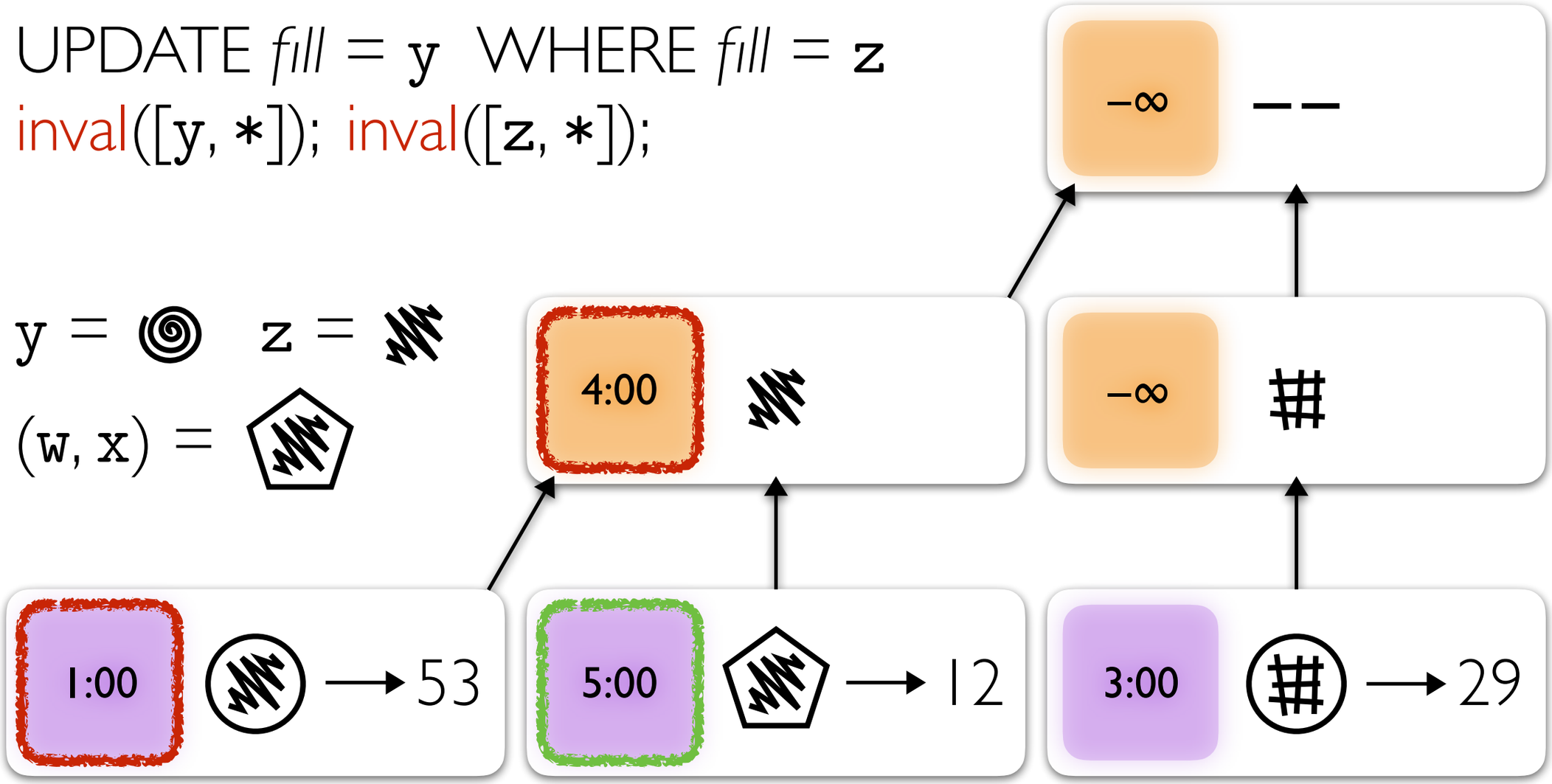
# Cache Data Structure

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`) =



Program instrumentation

SQL analysis

Cache data structure

Concurrency control

Program instrumentation

SQL analysis

talked

Cache data structure

talked

Concurrency control

Program instrumentation

Consolidate cached expressions, but avoid introducing new keys.

SQL analysis

talked

Cache data structure

talked

Concurrency control

Program instrumentation

Consolidate cached expressions, but avoid introducing new keys.

SQL analysis

talked

Cache data structure

talked

Concurrency control

Two global locks per cache: "data" lock and "transaction" lock.

Program instrumentation

Consolidate cached expressions, but avoid introducing new keys.

SQL analysis

talked

Cache data structure

talked

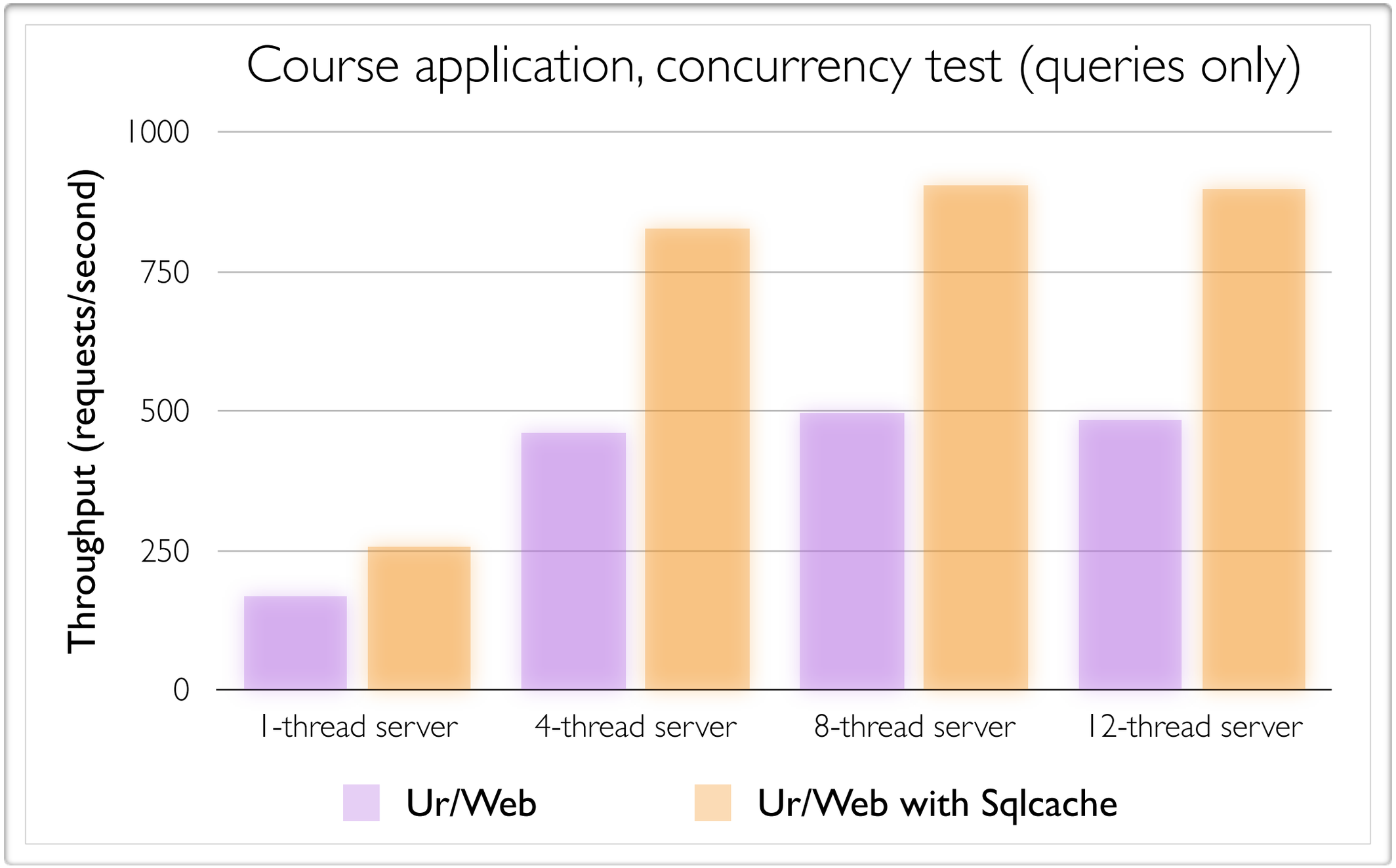
Concurrency control

Two global locks per cache: "data" lock and "transaction" lock.

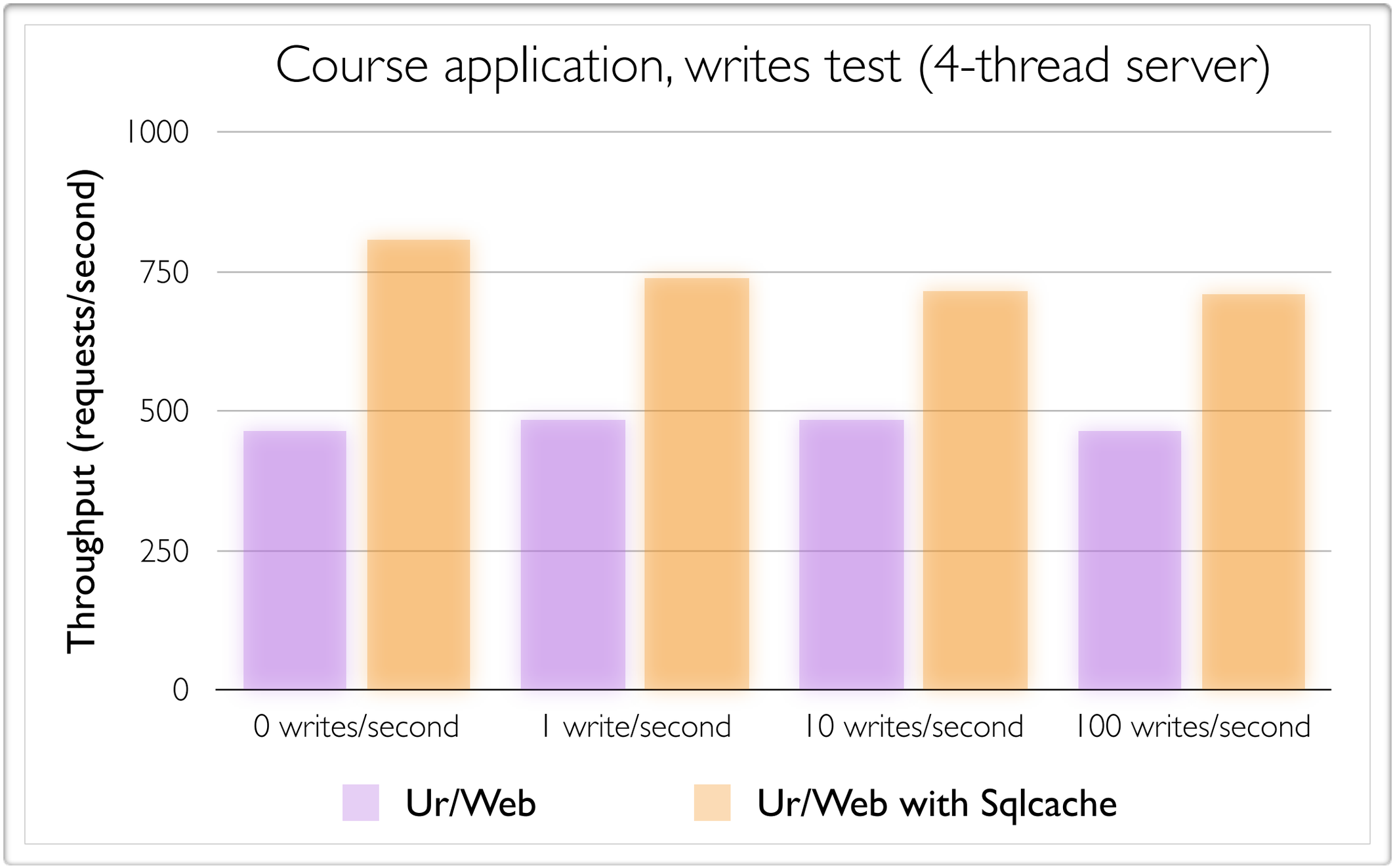
Runtime monitoring

Deactivate caches with low hit rate to reduce serialization.

# Performance Evaluation



# Performance Evaluation



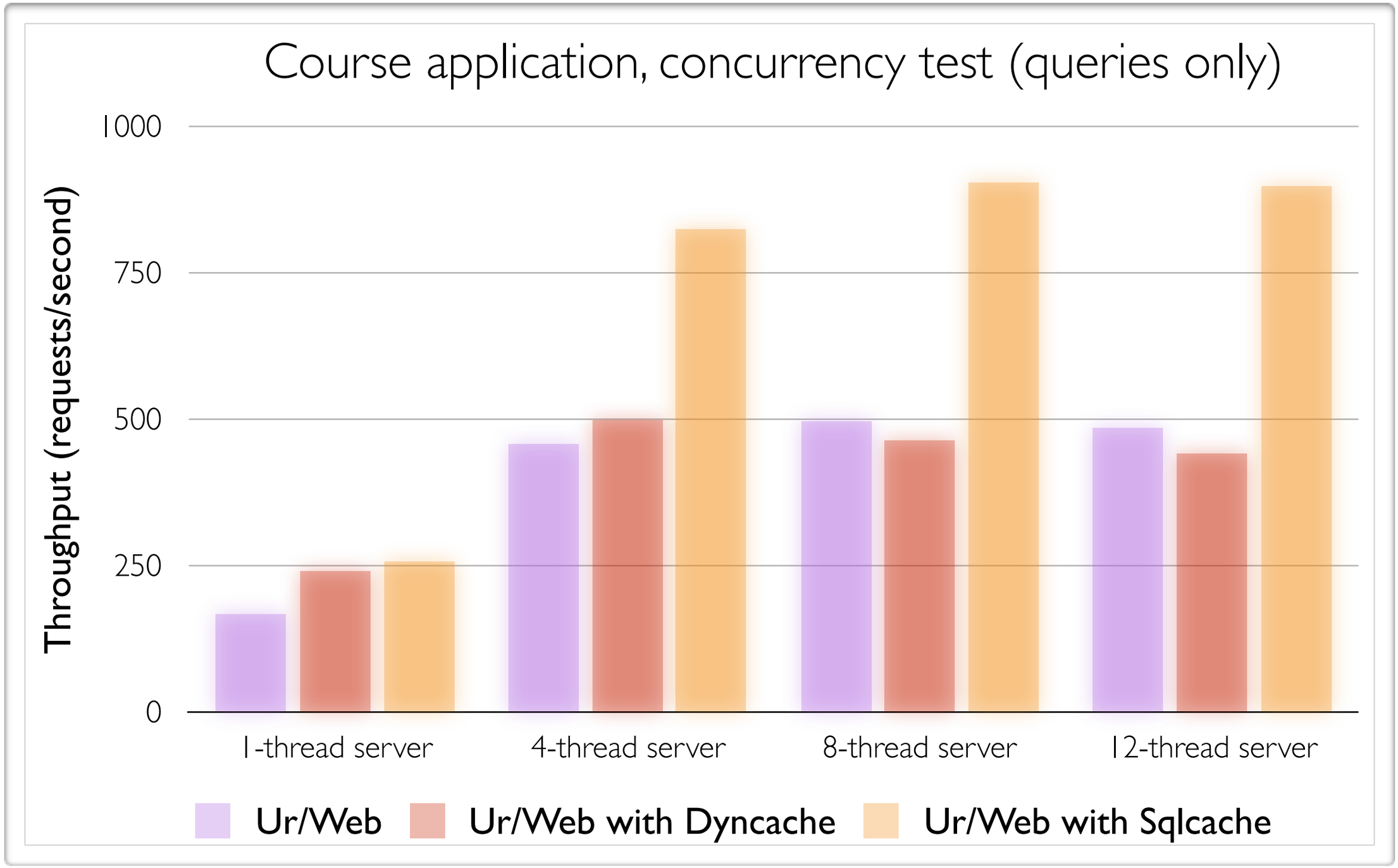
# Sqlcache

caching as a compiler optimization

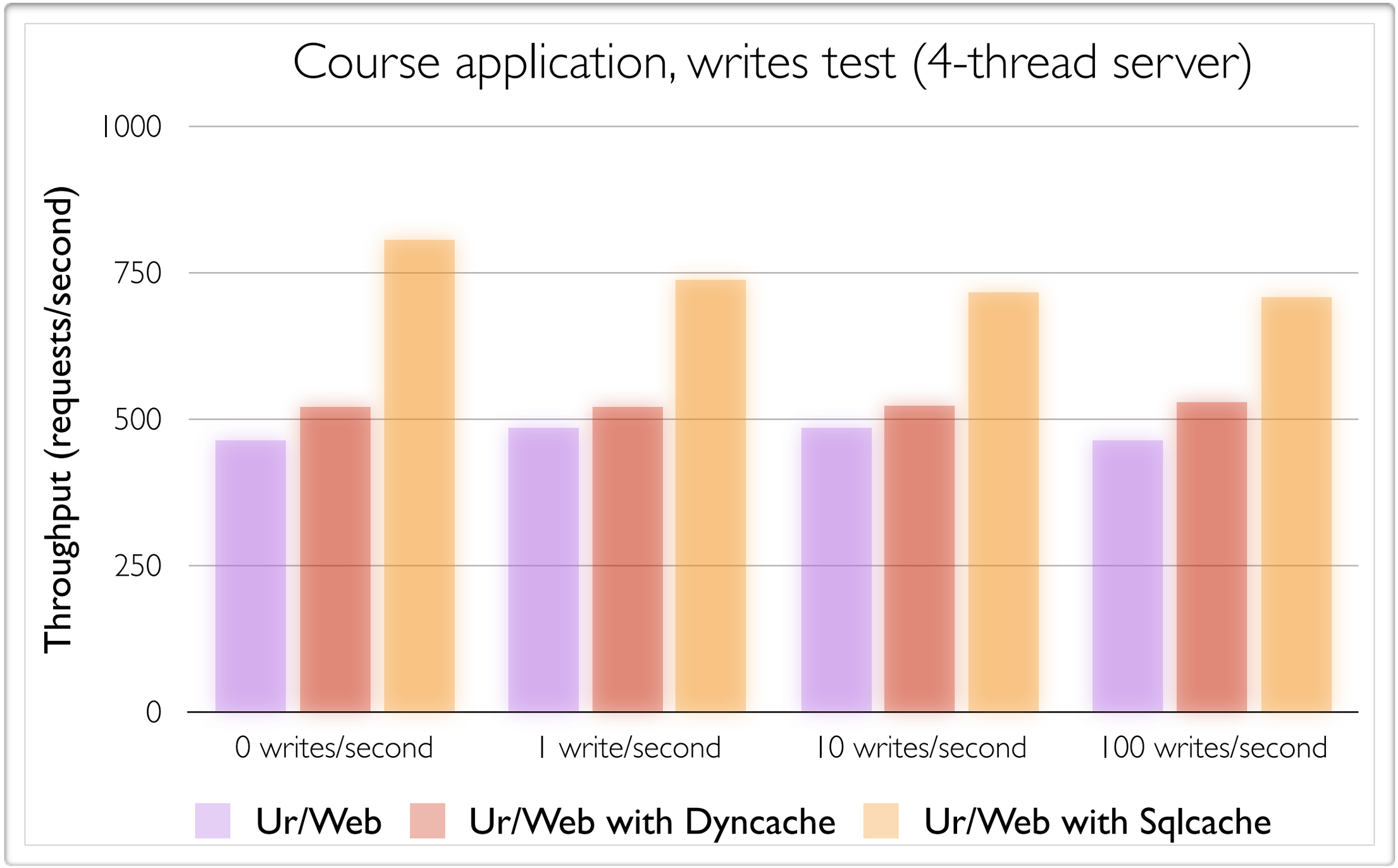
`https://github.com/urweb/urweb`

# Good question!

# Sqlcache vs. Dyncache



# Sqlcache vs. Dyncache



# 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?

```
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 adde

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

```
                    hape : int, Fill : int}

    =
    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 adde

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

First-class SQL

# Why Ur/Web?

```
                    hape : int, Fill : int}

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

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

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

Controlled side effects

Lots of inlining

First-class SQL