MySQL 101

Designing effective schema for InnoDB

Yves Trudeau

April 2015

MYSQL WORLDWIDE CONFERENCE & EXPO

PERCONA LIVE

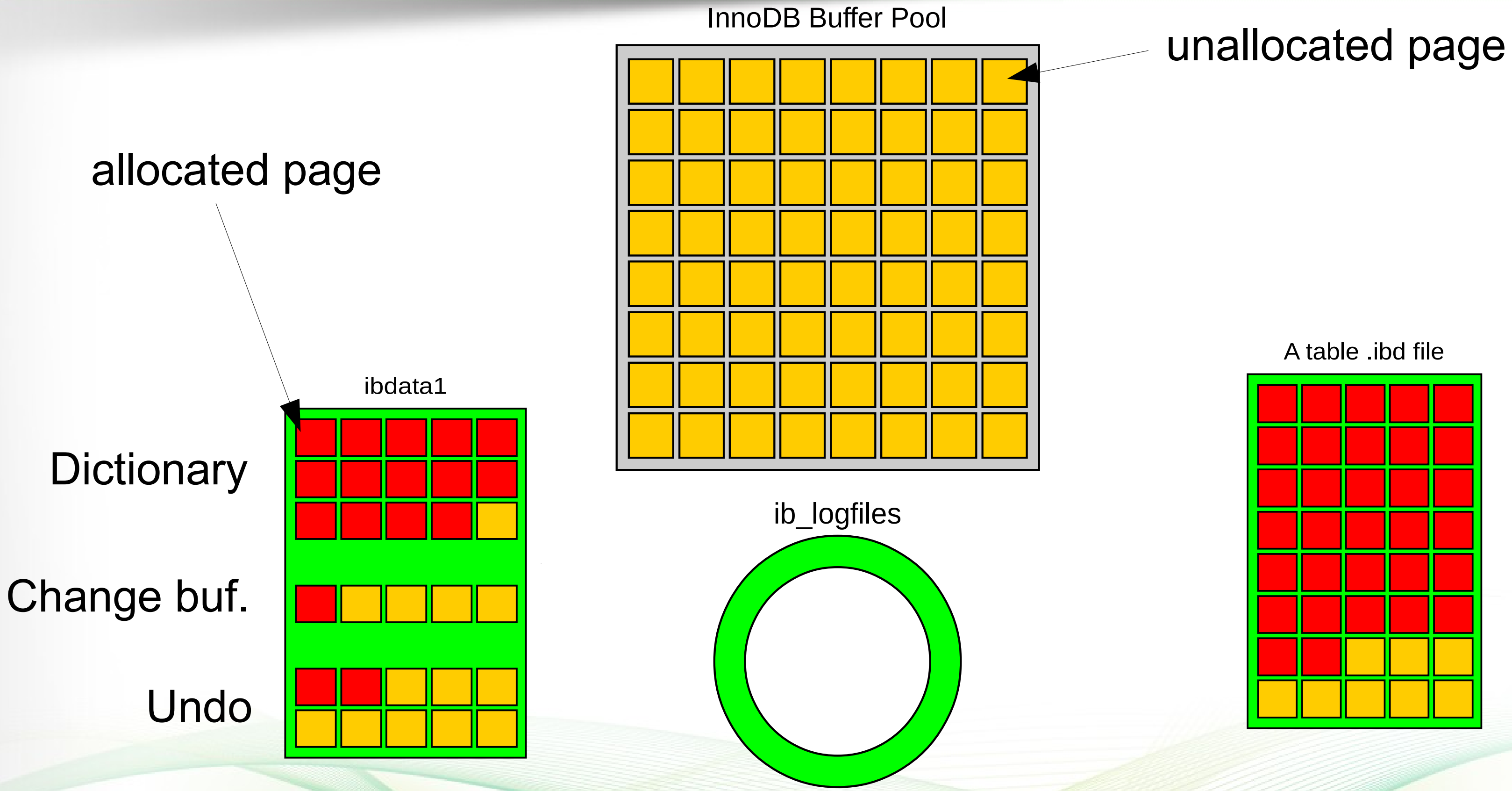
About myself : Yves Trudeau

- *Principal architect at Percona since 2009*
- *With MySQL then Sun, 2007 to 2009*
- *Focus on MySQL HA and distributed systems*
- *Database and science background*

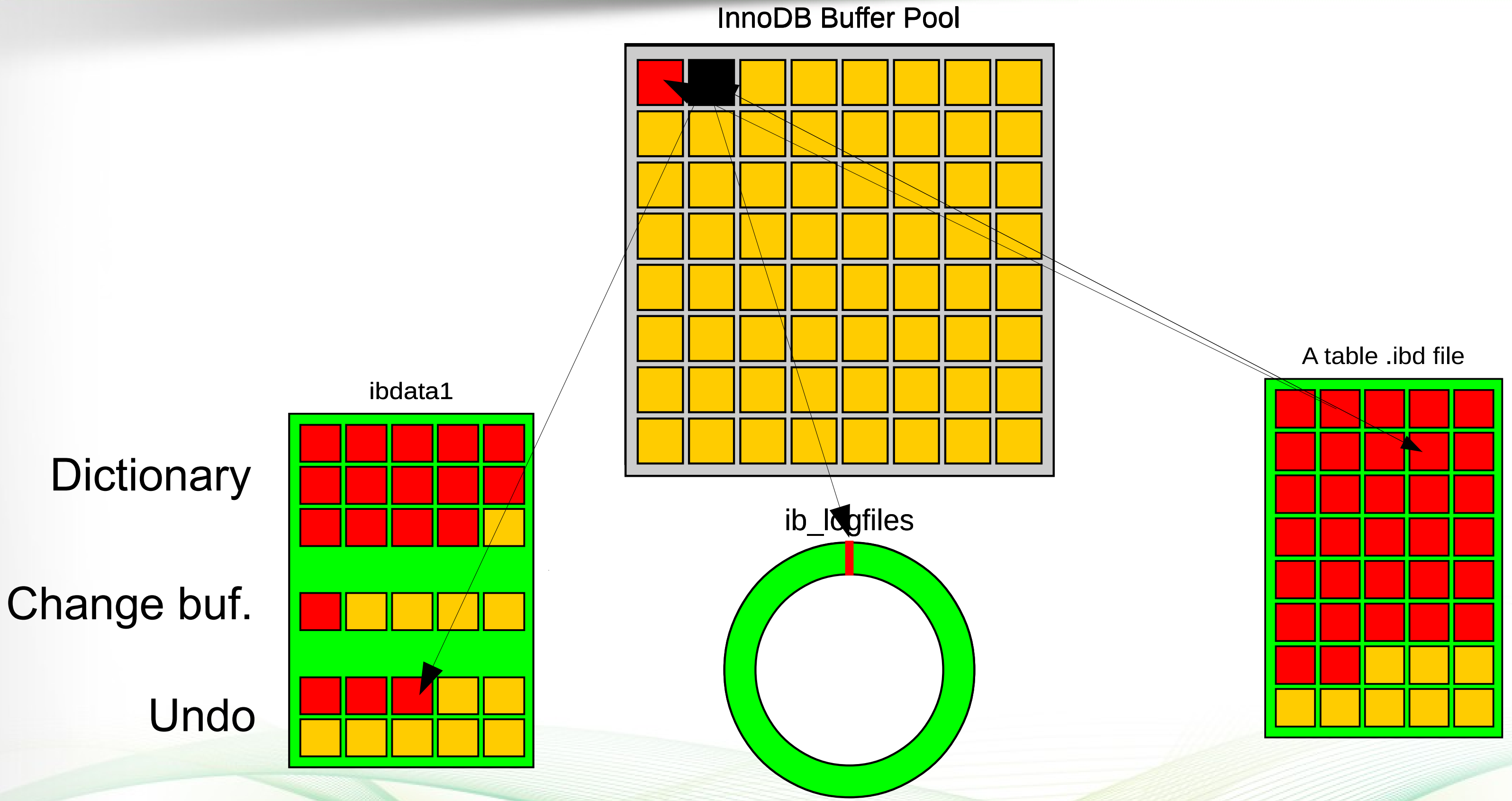
Plan

- *What's so special about InnoDB?*
- *Design of a web file sharing application*

A brief introduction to InnoDB Internals



Life cycle of an update query



Where's the data in InnoDB?

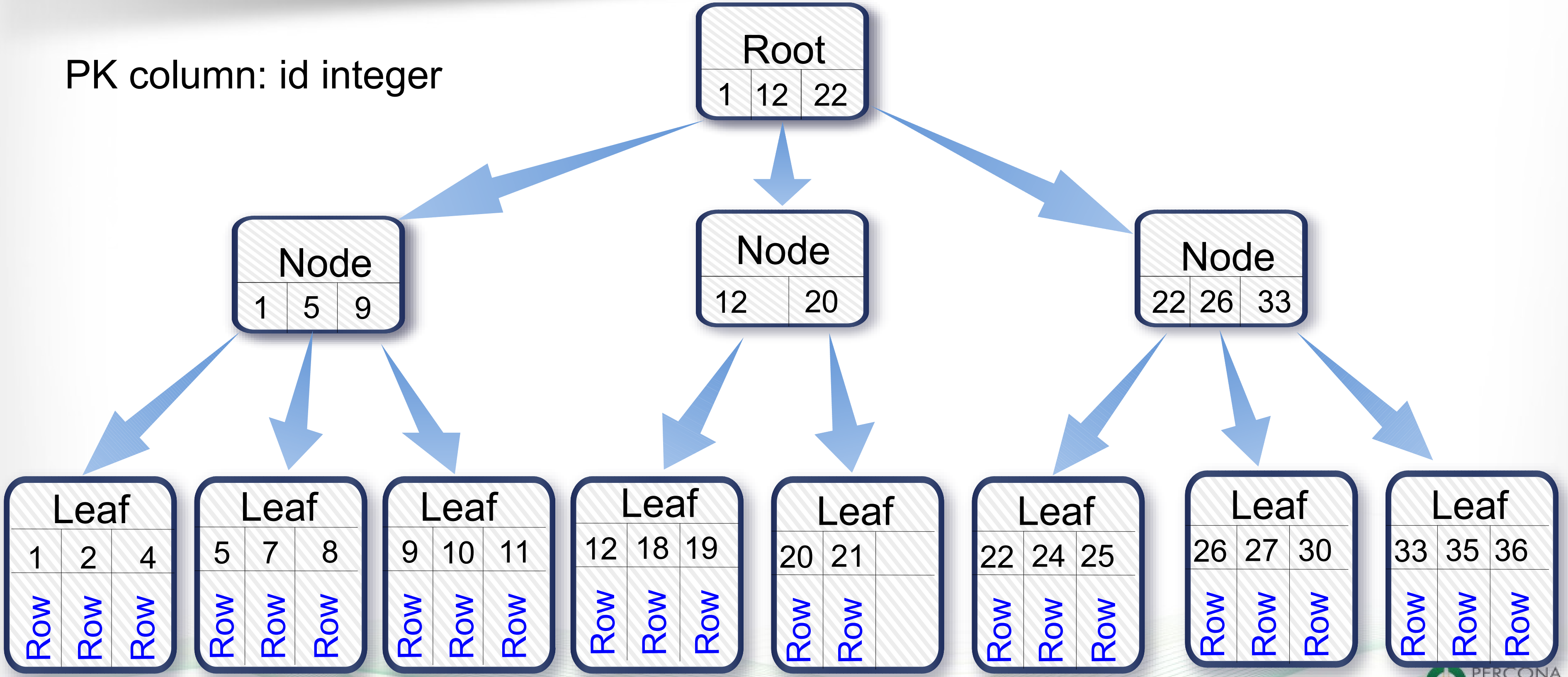
- *The rows are stored as values in the B-tree of the primary key*
- *The secondary keys store as values the primary keys of the matching rows*

Can't be true

I don't have PKs and it works!!

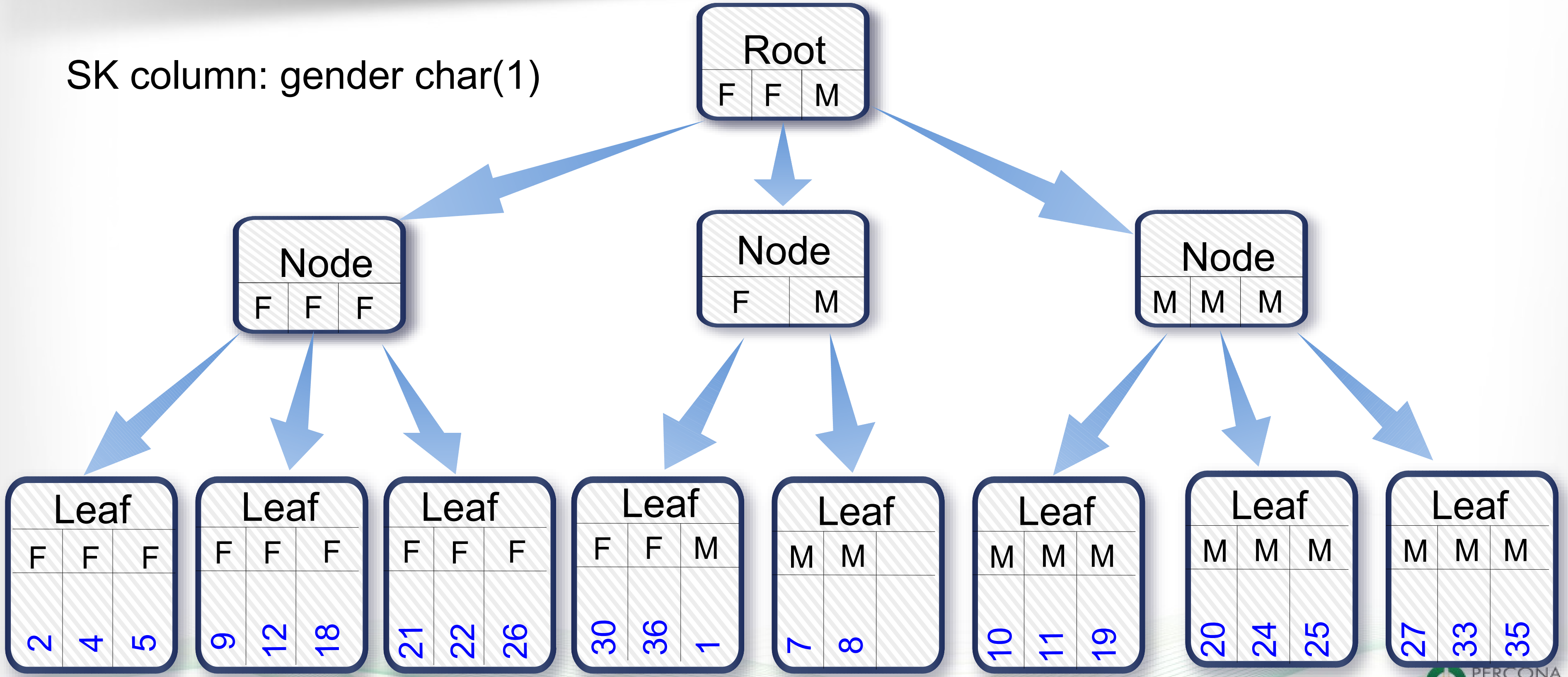
The primary key B-tree

PK column: id integer



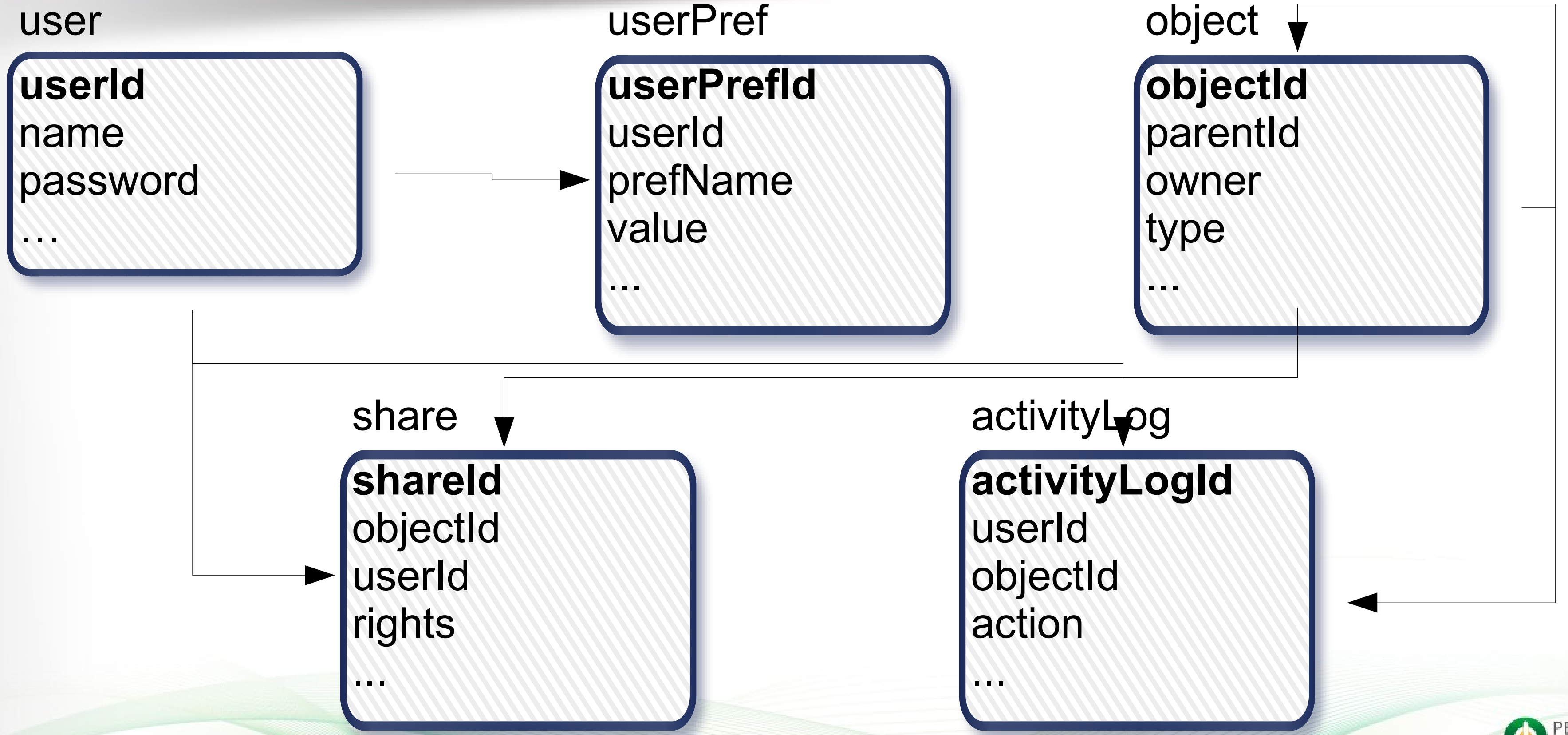
A secondary key B-tree

SK column: gender char(1)



Enough about InnoDB internals...

NewBox application, schema v1



NewBox application, table user

```
  CREATE TABLE `user` (
  `userId` char(36) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `password` char(32) DEFAULT NULL,
  `srvSchema` varchar(20) DEFAULT
NULL,
  `email` varchar(255) DEFAULT NULL,
  `updatedAt` datetime DEFAULT NULL,
  `createdAt` datetime DEFAULT NULL,
  `lastLogin` datetime DEFAULT NULL,
  `gender` char(1) DEFAULT NULL,
  PRIMARY KEY (`userId`),
  KEY `idx_name` (`name`),
  KEY `idx_password` (`password`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8
```

NewBox application, table userPref

```
CREATE TABLE `userPref` (
  `userPrefId` char(36) NOT NULL,
  `userId` char(36) DEFAULT NULL,
  `prefName` varchar(255) DEFAULT
NULL,
  `value` varchar(255) DEFAULT NULL,
  `updateddAt` datetime DEFAULT
NULL,
  `createdAt` datetime DEFAULT NULL,
  PRIMARY KEY (`userPrefId`),
  KEY `idx_userId` (`userId`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8
```

NewBox application, table object

```
CREATE TABLE `object` (
  `objectId` char(36) NOT NULL,
  `parentId` char(36) DEFAULT NULL,
  `ownerId` char(36) DEFAULT NULL,
  `type` varchar(20) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `urlStore` varchar(255) DEFAULT
NULL,
  `version` int(11) DEFAULT NULL,
  `deleted` int(11) DEFAULT NULL,
  `updatedAt` datetime DEFAULT NULL,
  `createdAt` datetime DEFAULT NULL,
  PRIMARY KEY (`objectId`),
  KEY `idx_owner` (`ownerId`),
  KEY `idx_name` (`name`),
  KEY `idx_urlStore` (`urlStore`),
  KEY `idx_deleted` (`deleted`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8
```

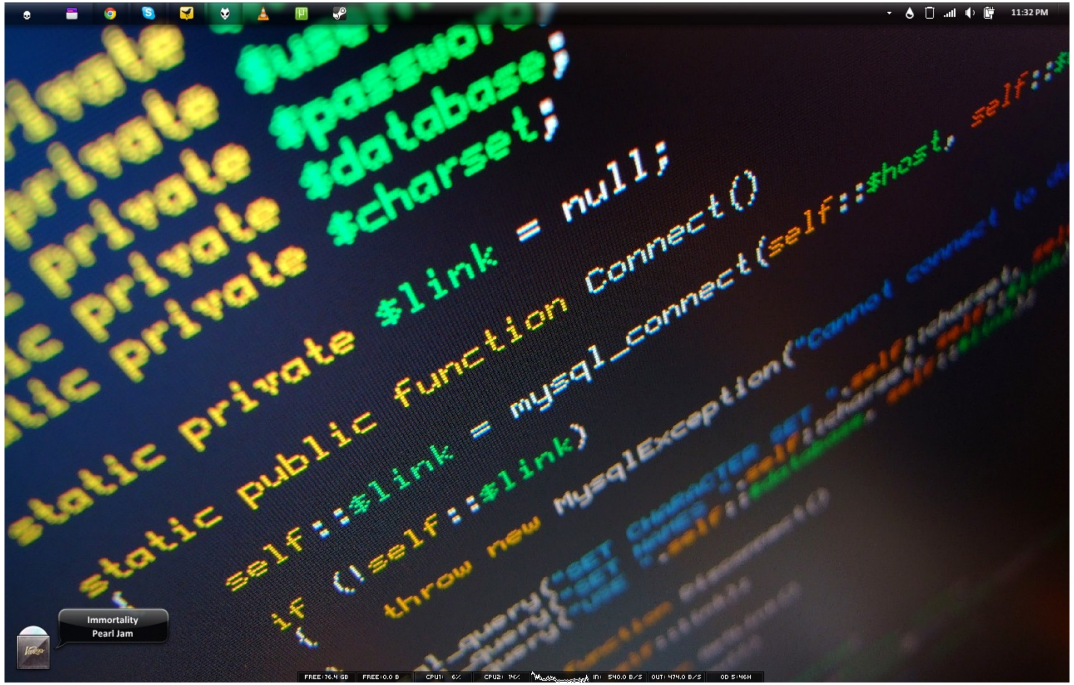
NewBox application, table share

```
CREATE TABLE `share` (
  `shareId` char(36) NOT NULL,
  `objectId` char(36) DEFAULT NULL,
  `userId` char(36) DEFAULT NULL,
  `ownerId` char(36) DEFAULT NULL,
  `rights` varchar(20) DEFAULT NULL,
  `updatedAt` datetime DEFAULT NULL,
  `createdAt` datetime DEFAULT NULL,
  PRIMARY KEY (`shareId`),
  KEY `idx_user` (`userId`)
) ENGINE=InnoDB DEFAULT
CHARSET=latin1
```

NewBox application, table activityLog

```
CREATE TABLE `activityLog` (
  `ActivityId` char(36) NOT NULL,
  `userId` char(36) DEFAULT NULL,
  `objectId` char(36) DEFAULT NULL,
  `action` varchar(255) DEFAULT NULL,
  `returnCode` int(11) DEFAULT NULL,
  `error` varchar(255) DEFAULT NULL,
  `IP` varchar(16) DEFAULT NULL,
  `createdAt` datetime DEFAULT NULL,
  PRIMARY KEY (`ActivityId`),
  KEY `idx_user` (`userId`),
  KEY `idx_object` (`objectId`),
  KEY `idx_created` (`createdAt`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8
```

NewBox application, Coding



```
rivate $user
private $passwor
private $database;
 private $charset;
ic private $link = null;
static private function Connect()
static public function Connect()
self::$link = mysql_connect(self::$host, self::
if (!self::$link)
throw new MysqlException("Cannot connect
```

NewBox application, pilot testing

NewBox application, stage load test

NewBox application, what's wrong?

- *Dataset is bigger than expected*
- *Database uses more CPU*
- *Database becomes slow when buffer pool is full*
- *Got lockings contention and even deadlocks!!!*
- *Disks are very busy*

NewBox application, what can we do?

- *More RAM?*
- *Faster drives?*
- *Shard earlier/more?*
- *Maybe my schema isn't that great...*

NewBox application, review of the data types

Importance of using the correct types

- *Optimal size = more data in cache*
- *Less reads and writes to disk*
- *Faster comparisons (less CPU)*

NewBox application, review of the data types

char with utf8

- *char type uses 3 bytes per char!!!*
- *uuid columns are thus char(108)*
- *keys on uuid columns with uuid pk are 216 bytes per entry*
- *change to varchar or use latin1 for the columns*

NewBox application, review of the data types

varchar with utf8

- *Why varchar(255)?*
- *a second length byte after 85*
- *Use proper length or stop at 85*

NewBox application, review of the data types

low cardinality columns

- *object.type → {file, folder, link}*
- *userPref.prefName → {theme, itemPerPage, defaultSort, etc}*
- *Use ENUM or a dictionary table*

NewBox application, review of the data types

Datetime

- *Arbitrary date and time*
- *8 bytes with 5.5.x, 5 bytes with 5.6.4+*
- *Timestamp ok for [1970,2036]*
- *Use timestamp*

NewBox application, review of the data types

Int types

- *Use the correct type → object.deleted tinyint*
- *No negative → unsigned*
- *bigint... is big*
- *using int unsigned for IPs (inet_aton and inet_ntoa functions)*

NewBox application, review of the data types

Blob/text types

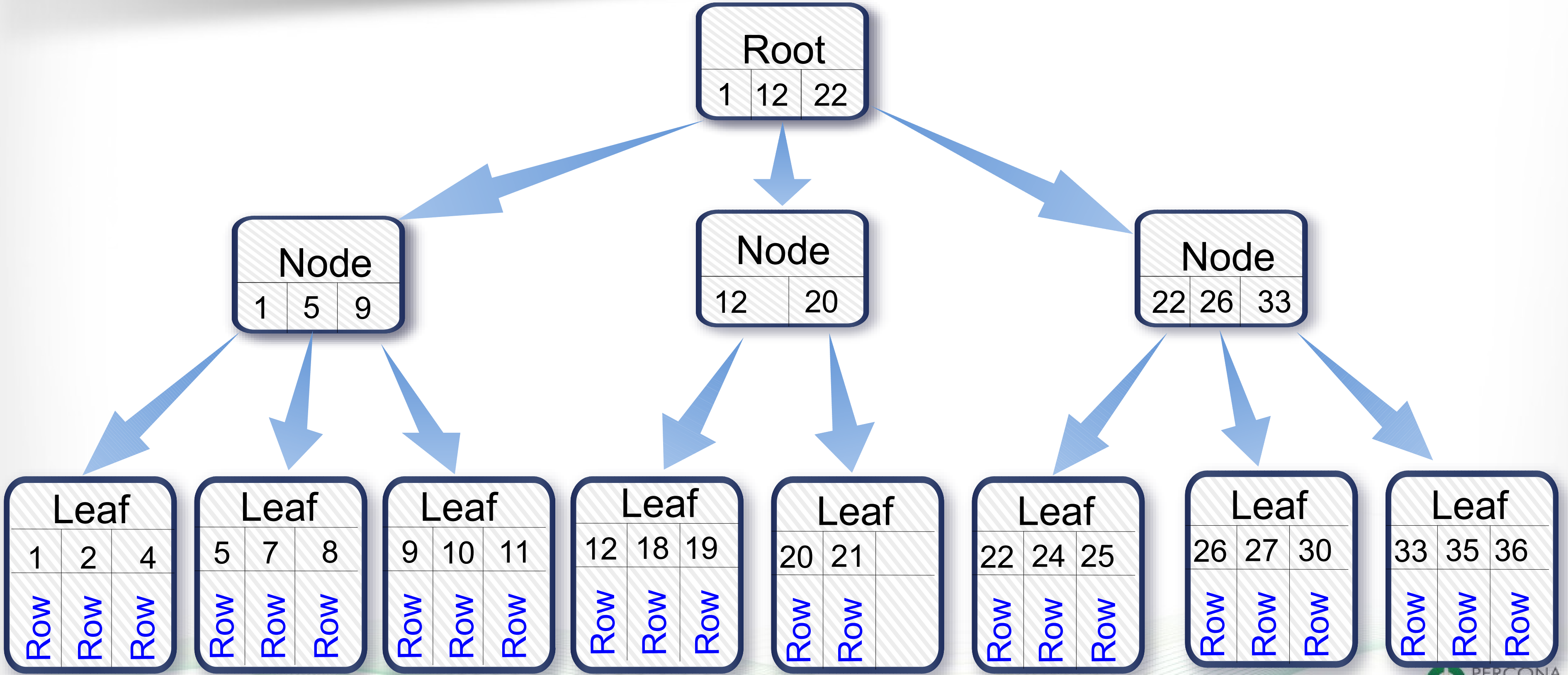
- *Split storage → overlay page*
- *More iops per row*
- *More on disk temp tables for queries (join/sort/group)*
- *Use compression if possible*

NewBox application, review of the data types

Is uuid a good thing?

- *Large varchar → slow to compare*
- *hex has low cardinality per byte*
- *inflate the size of the Sks*
- *random insert order*
- *Should use* **int unsigned auto_increment**

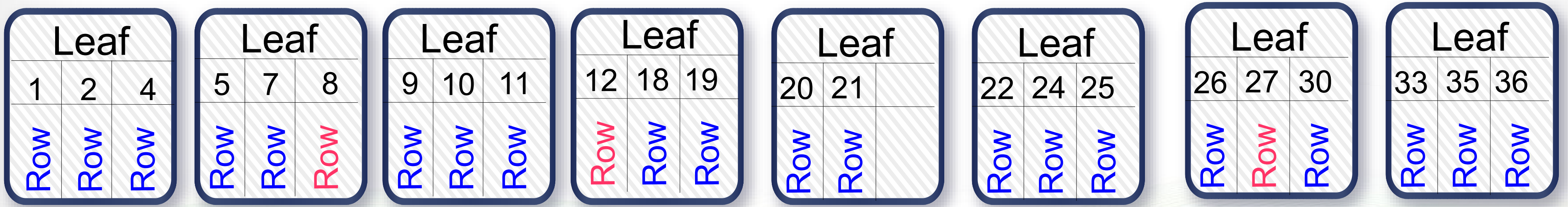
NewBox application, review of the PKs



NewBox application, review of the PKs

userPref, object, share

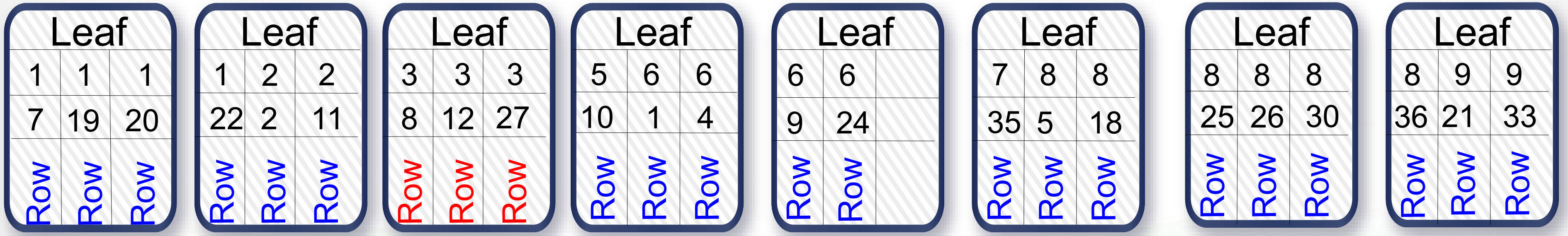
- *Retrieving object rows for a given userId or ownerId*
- *SKey on userId = 3 gives us: {8,12,27}*



NewBox application, review of the PKs

userPref, object, share

- *Reordering the Pks: objectId → UK*
- *PK → (userId,objectId)*



NewBox application, review of the PKs

activityLog

- *Lots of inserts*
- *Ok as auto_increment → merges writes*
- *minimize keys on master → use a slave*
- *Good idea to use partitions on ranges of activityLogId*

NewBox application, be sharding ready

Sharding

- *The ultimate scaling*
- *Start with 2 schema, NewBox_common and NewBox_data_1*
- *NewBox_common: { user, userPref }*
- *NewBox_data_1: { object, share, activityLog }*

NewBox application, indexing correctly

On large varchar

- *Slow to compare and big*
- *object.idx_name and object.idx_urlStore*
- *prefix issue with objstore, all start with 'http://'*
- *md5 hash?*
- *Better with a CRC32*

NewBox application, indexing correctly

Redundant keys

PRIMARY KEY (`userId`,`shareId`),
KEY `idx_user` (`userId`)

- *idx_user is useless, covered by the Primary key*
- *pt-duplicate-key-checker is your friend*

NewBox application, indexing correctly

Covering keys

Select o. from object o inner join share s on o.objectId = s.objectId where s.userId = 12345;*

- *idx_userId is there, not bad*
- *For each userId, needs to dive in s PK btree*
- *What about: (userId,objectId)*

NewBox application, indexing correctly

Index for sorting

Select o. from object where ownerId = 12345 order by createdAt;*

- *idx_owner is there*
- *Still has to sort the rows*
- *What if the key is : (ownerId,createdAt)*

NewBox application, indexing correctly

Over indexing...

- *No workload is the same*
- *Write intensive → be greedy on keys*
- *Read intensive → be generous on keys but careful not to harm cache*

NewBox application, indexing correctly

Tools

- *explain*
- *pt-query-digest*
- *Percona cloud tool*

Questions