MySQL 入門
COSCUP Hands-on 2015

Gea-Suan Lin
gslin@kkbox.com

我是誰？

我做過什麼事情？

今天的課程目的

Let’s start

Transaction

一組對資料庫的操作

有 ACID 特性

什麼是 ACID？

Principle of Transaction-Oriented Database Recovery

Theo Härder & Andreas Reuter

(1983)

實務上其實是個

商業名詞

Atomicity
(原子性)

原子不可分割性

舉例：銀行轉帳

```
BEGIN;
  SELECT * FROM account WHERE
  account_name = '陳啟祥' FOR UPDATE;

  UPDATE account SET money = money -
  83000000 WHERE account_name = '陳啟
  祥';

  UPDATE account SET money = money +
  83000000 WHERE account_name = '林益
  世';
COMMIT;
```

舉例：多筆資料的更新

```
UPDATE user
   SET user_valid = 0
 WHERE last_logined_at <
       '2015-01-01';
```

Consistency
(一致性)

```sql
CREATE TABLE account (
    id INTEGER,
    money INTEGER,
    CHECK (money >= 0)
);
```

MySQL 沒這東西

Isolation

(隔離性)

多個 Transaction 的交互行為

Durability
(持久性)

交易完成後保證資料的有效性

實務上會 Trade-off

大多數的情境下不會

用到完整的 A+C+I+D

Isolation 的四個等級

READ UNCOMMITTED
READ COMMITTED
REPEATABLE READS
SERIALIZABLE

READ

UNCOMMITTED

Dirty Reads

- Transaction 1

```
BEGIN;

SELECT age FROM users WHERE
id = 1; # 取得 “20”


SELECT age FROM users WHERE
id = 1; # 取得 “21”

COMMIT;
```

- Transaction 2

```
BEGIN;


UPDATE users SET age = 21
WHERE id = 1;


ROLLBACK;
```

https://en.wikipedia.org/wiki/Isolation_(database_systems)

READ COMMITTED

Non-repeatable reads

- Transaction 1

```
BEGIN;

SELECT age FROM users WHERE
id = 1; # 取得 “20”



SELECT age FROM users WHERE
id = 1; # 取得 “21”

COMMIT;
```

- Transaction 2

```
BEGIN;

UPDATE users SET age = 21
WHERE id = 1;

COMMIT;
```

https://en.wikipedia.org/wiki/Isolation_(database_systems)

REPEATABLE READS

Phantom Reads

- Transaction 1

```
BEGIN;

SELECT * FROM users WHERE
age BETWEEN 10 AND 30;

SELECT * FROM users WHERE
age BETWEEN 10 AND 30;

COMMIT;
```

- Transaction 2

```
BEGIN;

INSERT users (id, name, age)
VALUES (3, 'Bob', 27);

COMMIT;
```

https://en.wikipedia.org/wiki/Isolation_(database_systems)

SERIALIZABLE

總結 ACID

MySQL 上最常見的 ACID 方案

InnoDB

併購故事

Innobase 被 Oracle 買 (2005)

MySQL 被 Sun 買

(2008)

Sun 被 Oracle 買
(2010)

Oracle Makes Commitments to Customers, Developers and Users of MySQL
(Oracle Press Release, 2009/12/14)

Percona

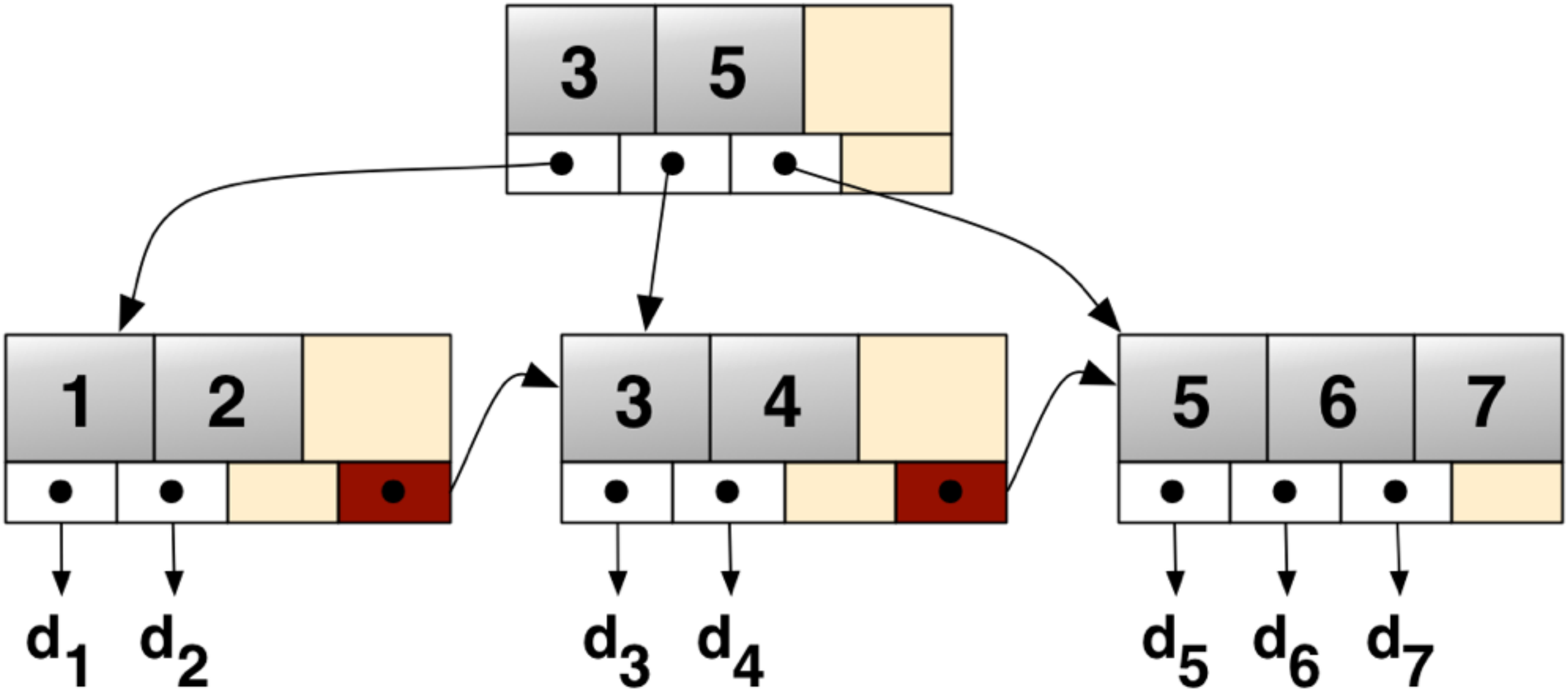
MariaDB

選擇

講 Index 前

資料結構

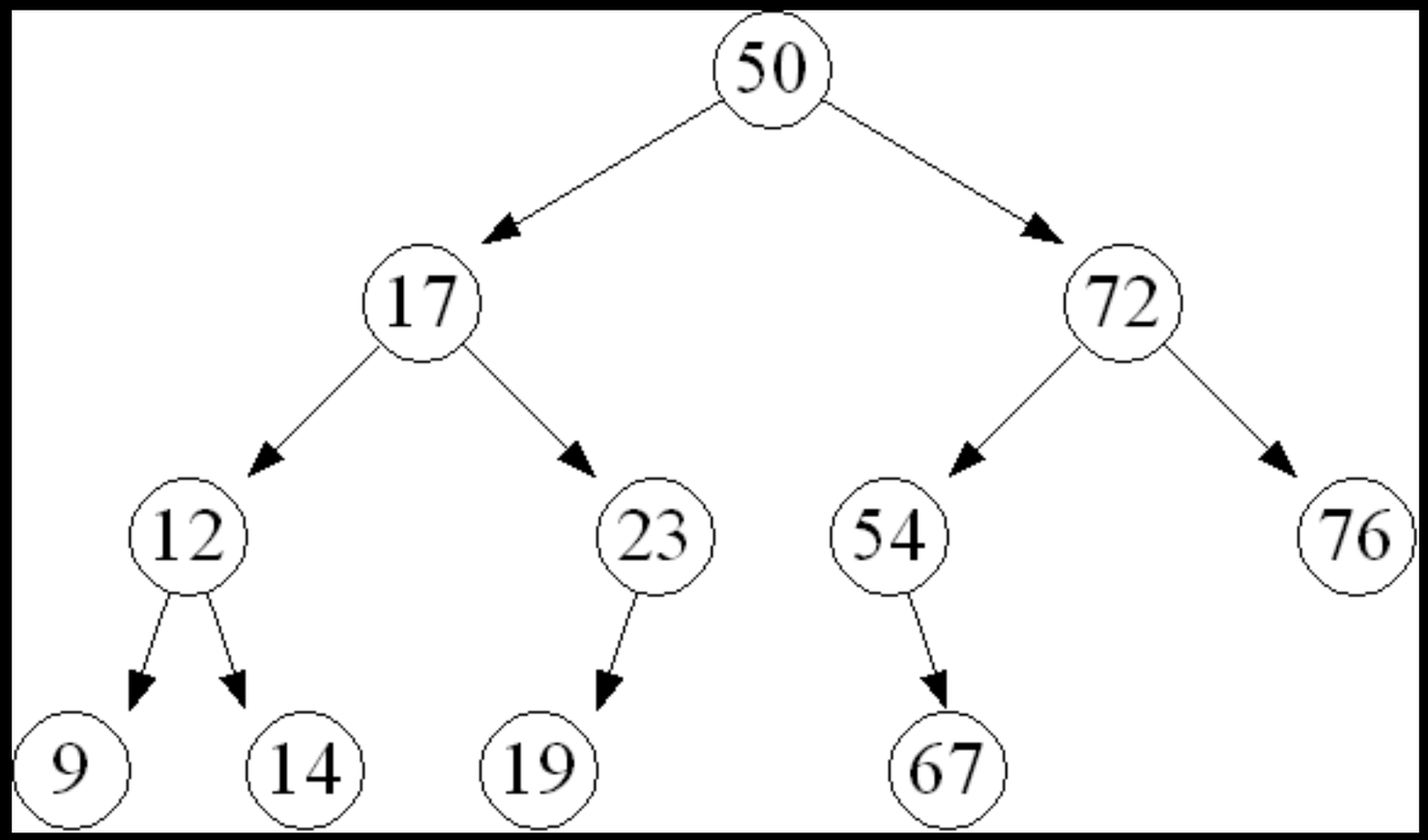
B+ Tree



https://en.wikipedia.org/wiki/B%2B_tree

Self-Balancing Binary Search Tree

AVL、Red-Black Tree



https://en.wikipedia.org/wiki/File:AVLtree.png

B+ Tree 的特性

Linear Order

很重要

Linear Order
Linear Order
Linear Order

定義 Index 就是

定義 Order

為什麼不使用

Binary-based Tree ?

深度的差異與

Random Seek 次數

即使是 SSD

也還是很重要

Index

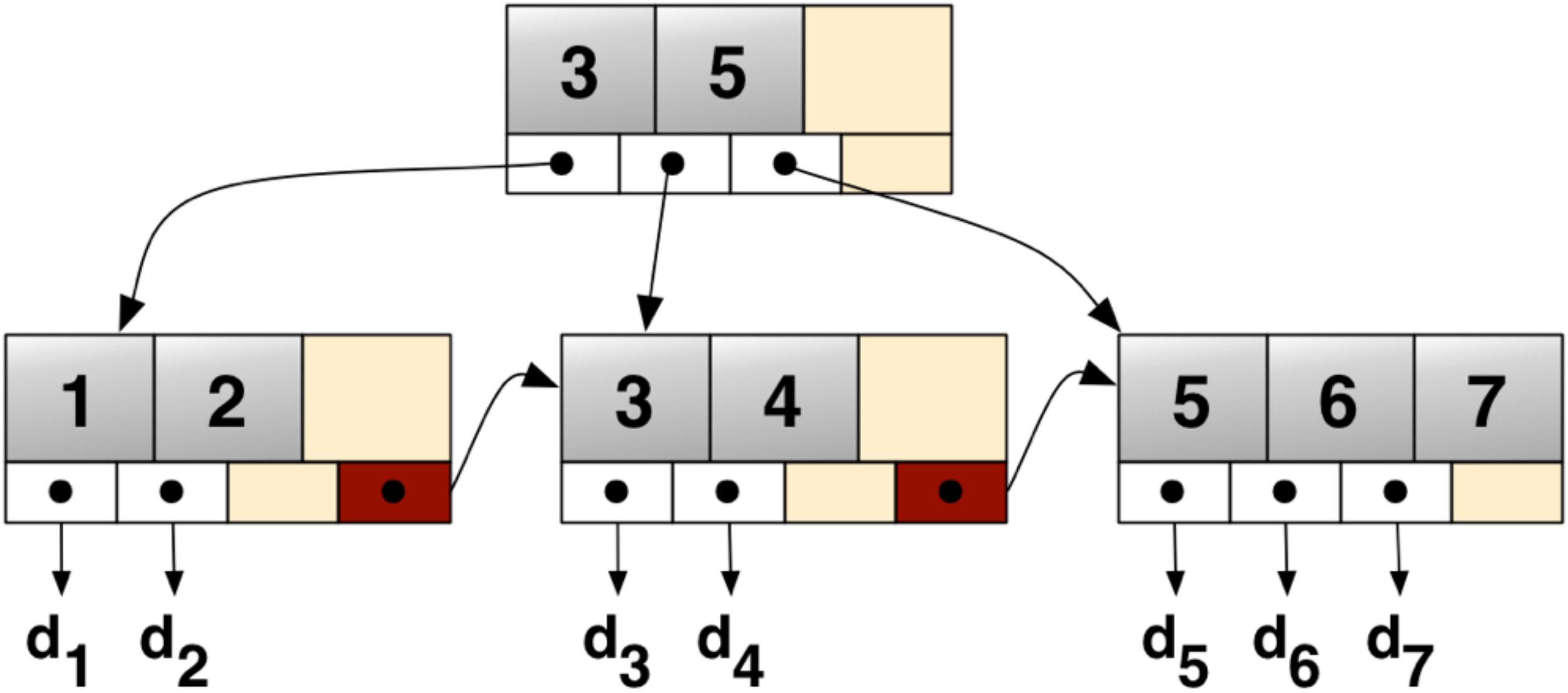
Primary Key

Secondary Index

PRIMARY KEY

避免 Compound PK

Clustered Index

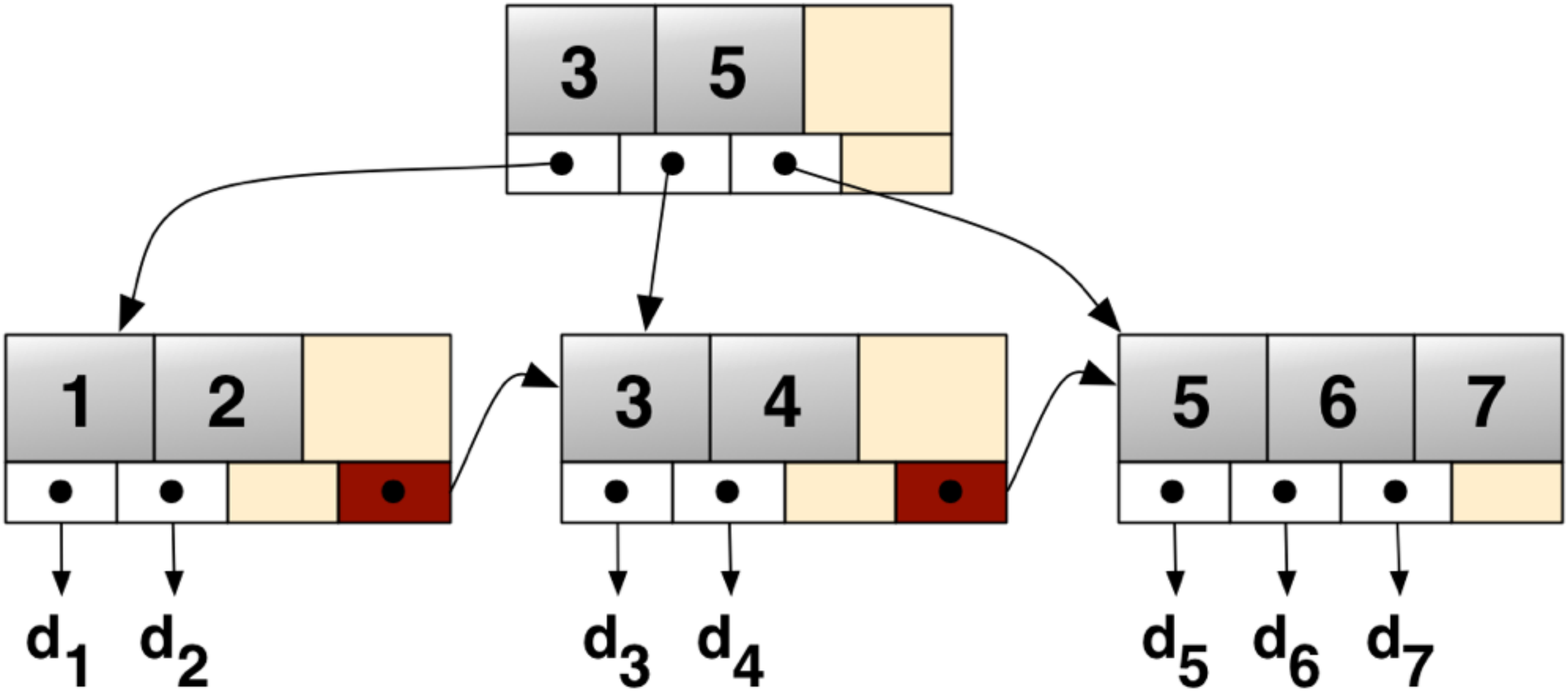


https://en.wikipedia.org/wiki/B%2B_tree

Primary Key 的
Range Search 很快

Secondary Index

指到 PRIMARY KEY



https://en.wikipedia.org/wiki/B%2B_tree

用 Secondary Index
查詢比較慢

UNIQUE KEY

可以 NULL

因為要確保唯一

需要 Lock

INDEX KEY

多加通常不會有用

常見錯誤範例

表格內有多少欄位就設多少：
(col_a)
(col_b)
(col_c)

範圍重複的 Index：

(col_a, col_b, col_c)

(col_a, col_b)

(col_a)

怎麼才是正確的

Index ?

依照需求而設計

High Availability

Master + Slave

Async Replication

STATEMENT 與 ROW

Replication Lag

利用 ACID 中的 D

Crash-safe

DRBD + Heartbeat

DRDB

Network-based RAID 1

Heartbeat
偵測機制

Floating IP
(Virtual IP)

Q&A

(然後實作)