IRON
HACK

SQL Indexes

Andreas Finger
@mediafinger

SQL

- Structured Query Language
- used by RDBMS (Relational DataBase Management Systems)
  e.g. MySQL, PostgreSQL, Oracle
- not used by noSQL databases ;-)
  e.g. Redis, CouchDB, MongoDB

SQL

- In Rails we can formulate most queries with ActiveRecord
- but sometimes you will need SQL in Rails
- SQL can be much(!) faster than ActiveRecord
- and often you want to quickly get some data out of your database
  e.g. run some statistics, export some tables

SQL

- Task.where(status: "todo", user_id:1)

  ```
  SELECT * FROM tasks
  WHERE status = ‘todo’
  AND user_id = 1
  ```

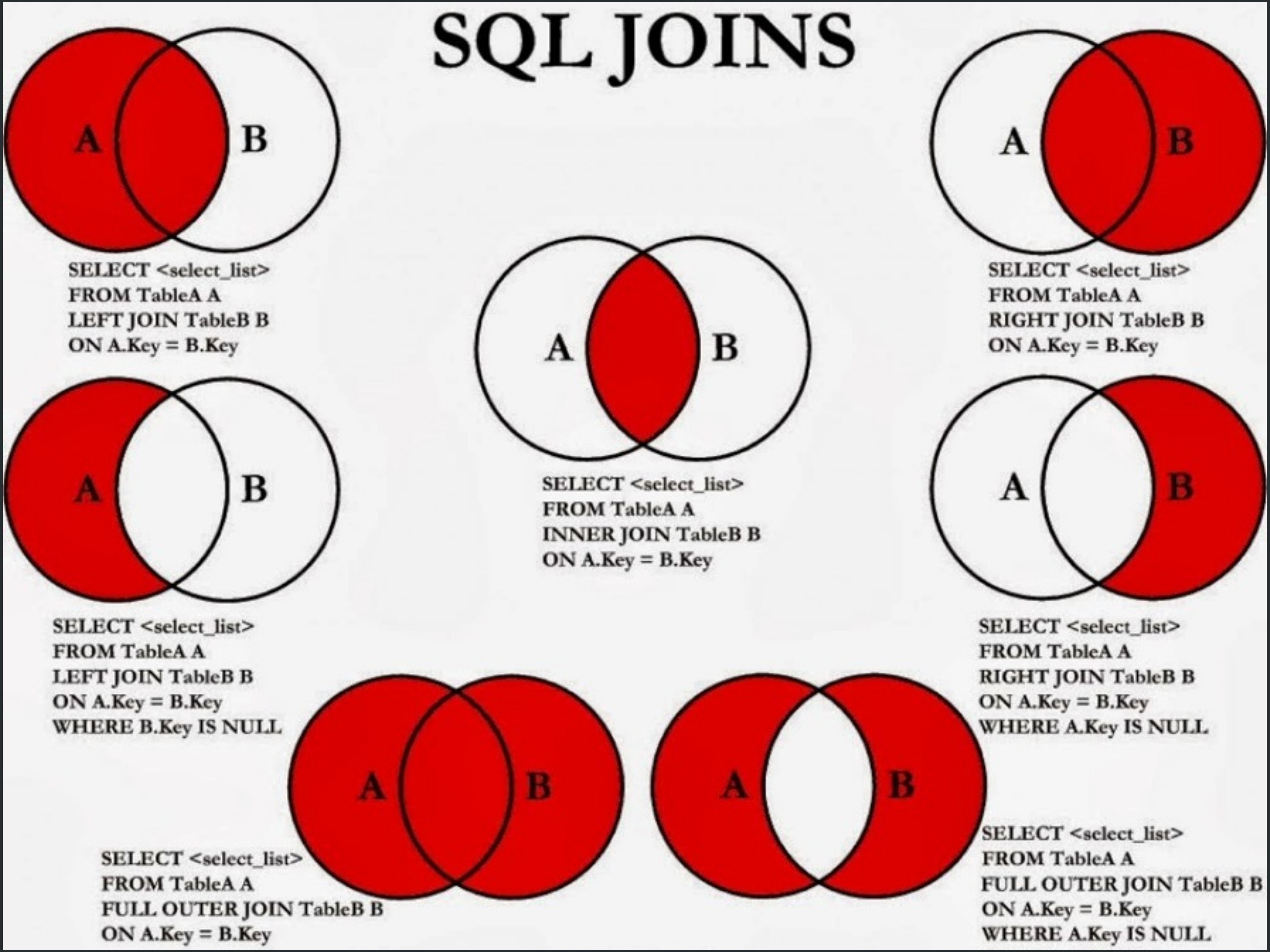
- Task.where(status: [“todo”, “doing], user_id:[1,2]).
  .order(updated_at: :asc)

  ```
  SELECT * FROM tasks
  WHERE status IN ('todo', 'doing')
  AND user_id IN (1,2,3,4)
  ORDER BY updated_at ASC
  ```

SQL

- Project.first.users
  ```
  SELECT users.* FROM users
  INNER JOIN projects_users ON users.id =
  projects_users.user_id
  WHERE projects_users.project_id = 1
  ```

- User.where(confirmed: true).
  joins(:tasks).where(:tasks, status: ["todo", "doing"])
  ```
  SELECT users.* FROM users
  INNER JOIN tasks ON tasks.user_id = users.id
  WHERE users.confirmed = 1
  AND tasks.status IN ('todo', 'doing')
  ```



SQL JOINS

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

SQL Indexes

- You should create an Index, when you
  - access elements on a table often with foreign keys e.g. Tasks.where(user_id: 1)
  - often use queries that join tables
  - sort by a large dataset by a column often

SQL Indexes

- When you create an Index, SQL needs some time to go through the data, sort it and write the Index to disk
- That means every Index costs disk space
- Indexes that cover several fields, are harder to create the right way, as the order of the fields matter
- You only need an Index if you have A LOT of data as RDBMS are made to handle many millions of entries

SQL Indexes

- Indexes can speed up queries a lot

- For large datasets and complex operations it could look like this:
  30 seconds (no index)
  4 seconds (a not perfect index)
  0.08 seconds (the right index)

- any normal query should just take a few milliseconds

<

- Databases are optimized to search, join, aggregate, group and order data

- They can also do caculations much faster than it could be done on the server side

- Whenever possible, let your database do the heavy lifting and just work with the results

noSQL

- Key-Value Stores are a type of noSQL databases
- Key-Value-Stores are often used for Caching, or to save the messages of a Queue
- Key-Value Stores are very fast, because they are very simple
- A popular Key-Value Store is Redis
- Postgres has a Key-Value Store build in: HSTORE

noSQL

- Document Stores are a type of noSQL databases
- DS excel at saving complete datasets without fixed form or size and allowing to access them
- DS usually have good full text search capabilities
- DS have are not great when running reporting over the whole database
- the flexibility of DS looks great at the beginning, but can become technical debt later