カタログデータベース検索

峯尾聡吾

目次

- データベースとSQL
  - データベース検索のための汎用の知識

- HSC-SSP データベースについて

データベースとSQL

- 多くの「テーブル」を検索できるようにしたもの
  = **データベース**
- 検索するための言語 = **SQL**

(↓テーブルの一例)

id	ra	dec	mag	mag_err	failure
1000	150.0	2.0	25.0	1.0	true
1001	150.1	2.1	25.1	1.1	true
1002	150.2	2.2	25.2	1.2	false
1003	150.3	2.3	25.3	1.3	false
1004	150.4	2.4	25.4	1.4	true
1005	150.5	2.5	25.5	1.5	true
1006	150.6	2.6	25.6	1.6	false

結論から先に (前半部)

- 詳しくは「SQL　入門」でグーグル先生に聞く

- SQL には方言が少しある。
  - 入門サイトの例文がエラーになるときは
    方言の問題かもしれない。
  - 「PostgreSQL」を検索ワードに入れると
    解決するかもしれない。

テーブル全部を取得

- `SELECT * FROM catalog;`

catalog

id	ra	dec	mag	mag_err	failure
1000	150.0	2.0	25.0	1.0	true
1001	150.1	2.1	25.1	1.1	true
1002	150.2	2.2	25.2	1.2	false
1003	150.3	2.3	25.3	1.3	false
1004	150.4	2.4	25.4	1.4	true
1005	150.5	2.5	25.5	1.5	true
1006	150.6	2.6	25.6	1.6	false

テーブル全部を取得

- `SELECT * FROM catalog;`

- 大文字・小文字は×区別しない。
- 文字列値はシングルクォーテーションでくくる
  - ‘Abc’（文字列値は大文字小文字を**区別する**）
- ダブルクォーテーションは

×文字列値の意味ではない。普通は使わない。

用語

- クエリ (Query /ˈkwɪəɹi/, /ˈkwɛɹi/)
  - 「問い合わせ」 (Question と同根。cf. inquiry)
  - だいたい SQL 文のことを指して使う
- コラム (Column)
  - 「柱」の意味
  - テーブルの「縦の列」を指す。「カラム」と書く人もいる
- 行 (Row)
  - 「ロー」とは日本語であまり言わない気がする
  - テーブルの「横の行」を指す

コラム (縦の列) を選択

- `SELECT ra,dec,mag FROM catalog;`

catalog

id	ra	dec	mag	mag_err	failure
1000	150.0	2.0	25.0	1.0	true
1001	150.1	2.1	25.1	1.1	true
1002	150.2	2.2	25.2	1.2	false
1003	150.3	2.3	25.3	1.3	false
1004	150.4	2.4	25.4	1.4	true
1005	150.5	2.5	25.5	1.5	true
1006	150.6	2.6	25.6	1.6	false

行 (横の行) を選択

- `SELECT * FROM catalog`
  `WHERE NOT failure;`

catalog

id	ra	dec	mag	mag_err	failure
1000	150.0	2.0	25.0	1.0	true
1001	150.1	2.1	25.1	1.1	true
1002	150.2	2.2	25.2	1.2	false
1003	150.3	2.3	25.3	1.3	false
1004	150.4	2.4	25.4	1.4	true
1005	150.5	2.5	25.5	1.5	true
1006	150.6	2.6	25.6	1.6	false

テーブルの結合

- 各生徒を担当する担任が知りたい

(このあたりから SQL が楽しくなります)

studentList

student	class
はな	C組
ひまり	B組
あかり	A組
いちか	C組

classList

class	担任
A組	佐藤
B組	鈴木
C組	高橋

テーブルの結合

- 各生徒を担当する担任が知りたい
- SELECT * FROM studentList JOIN classList ON (studentList.class = classList.class);

studentList

student	class
はな	C組
ひまり	B組
あかり	A組
いちか	C組

classList

class	担任
A組	佐藤
B組	鈴木
C組	高橋

studentList JOIN classList ON (…)

student	class		class	担任
はな	C組	=	C組	高橋
ひまり	B組	=	B組	鈴木
あかり	A組	=	A組	佐藤
いちか	C組	=	C組	高橋

結合 (JOIN) の詳しい説明

- **テーブル**とは行の**集合**である:
  - A = { 行1, 行2,… }, 行 = (値1,値2,…)
  - B = { 行1, 行2,… }
- **結合**とは集合の**直積**である:
  - A join B on(条件)

    = { (行a, 行b) | 行a∈A かつ行b∈B かつ 条件 }
- ややこしい結合で混乱したときはここに立ち返るとよい

結合条件の書き方

- SELECT * FROM
  studentList JOIN classList
  ON (studentList.class=classList.class);

- 个面倒くさいので下のように略記可能↓

- SELECT * FROM
  studentList JOIN classList
  USING (class);

結合は慎重に (重要)

- 結合条件によっては答えが返ってこなくなる。
  - 検索に時間がかかりすぎるため
  - 検索用の索引(インデクス)が付いているコラムを使うのが良い
- 結合条件を満たさない行は静かに消える。
  - 次ページ

結合の種類

- もし studentList に「D組」の生徒がいるのに classList に「D組」が抜けている場合
  その生徒は居なかったことになる

studentList

student	class
はな	C組
ひまり	B組
あかり	A組
いちか	C組
さくら	D組

classList

class	担任
A組	佐藤
B組	鈴木
C組	高橋

D組の情報がない!

studentList JOIN classList ON(…)

student	class		class	担任
はな	C組	=	C組	高橋
ひまり	B組	=	B組	鈴木
あかり	A組	=	A組	佐藤
いちか	C組	=	C組	高橋

~~「さくら」~~のレコードが抹消される

左結合

- JOIN の左オペランドを抹消したくない場合は LEFT JOIN を使う (同様に RIGHT JOIN も FULL JOIN もある)
- `studentList LEFT JOIN classList ON (studentList.class = classList.class);`

studentList

student	class
はな	C組
ひまり	B組
あかり	A組
いちか	C組
さくら	D組

classList

class	担任
A組	佐藤
B組	鈴木
C組	高橋

studentList **LEFT** JOIN classList ON(…)

student	class		class	担任
はな	C組	=	C組	高橋
ひまり	B組	=	B組	鈴木
あかり	A組	=	A組	佐藤
いちか	C組	=	C組	高橋
さくら	D組		NULL	NULL

三つ以上の結合

- `A join B on (A.a = B.b)`

は全体で一つのテーブルを表すので

- `(A join B on (A.a = B.b))`
  `join C on (A.a = C.c)`

のように書けば三つのテーブルを結合できる。

かっこを外して

- `A join B on (A.a = B.b)`
  `join C on (A.a = C.c)`

とも書ける

別名のつけ方

- コラムやテーブルに別名をつけられる。
- SELECT

```
    x.mag AS magnitude,
    y.z   AS redshift
FROM
    catalog      AS x
    JOIN photo_z AS y
        ON (x.id = y.objid)
WHERE
    x.mag < 25 AND NOT x.failure
;
```

クエリの診断

- クエリが返ってこないなあ…と思ったら、explain 文を投げてみてください。

- EXPLAIN SELECT …(説明がほしいクエリ)…;

↓回答

一つめのレコードが返るまでにかかる時間

すべてのレコードが返るまでにかかる時間

(数値の単位は 8KB をストレージから読み込む時間)

- Nested Loop Left Join
  (cost=26749.27..5532931.67 rows=86650 width=2331)

HSC-SSP DATA RELEASE

存在するテーブルについて

- 詳しくは↓へ

https://hscdata.mtk.nao.ac.jp/schema_browser2/

テーブルの大別

- Metadata (“data about data”)
  - 画像(data)にかかわる情報(data)
  - 撮像日時、望遠鏡の向いていた方向、など
  - 小さなテーブルなのでおざなりに使ってよい

- Catalog
  - 天体の情報
  - 位置、等級、形状、など
  - 巨大なテーブルなので注意して使う

Metadata

- Frame
  - 撮像された個別のCCD画像一枚一枚の情報
- Mosaic__deepcoadd
  - スタックの Patch画像一枚一枚の情報
- Mosaicframe__deepcoadd
  - 「ある Frame が ある Mosaic に含まれているか」という情報
- Warped__deepcoadd
  - ワープ (座標変換) のみを行った、足し合わせる前の画像
- Fcr
  - Frame の Flux Correction (モザイクによって決まったもの)
- Wcs
  - Frame の Wcs (モザイクによって決まったもの)

Metadata

- Frame
  - 撮像された個別のCCD画像一枚一枚の情報
- Mosaic__deepcoadd_?
  - スタックの Patch画像一枚一枚の情報
- Mosaicframe__deepcoadd_?
  - 「ある Frame が ある Mosaic に含まれているか」という情報
- Warped__deepcoadd_?
  - ワープ (座標変換) のみを行った、足し合わせる前の画像
- Fcr
  - Frame の Flux Correction (モザイクによって決まったもの)
- Wcs
  - Frame の Wcs (モザイクによって決まったもの)

接尾辞の意味

- **__deepcoadd**
  - 「深いスタック」の意味。当初はいろんなスタック画像を用意する腹積もりだったが現状はこれしかない
- **__merged** (次に出てくる)
  - 「merged カタログをレファレンスにして Forced photometry した」の意味。当初はバンドごとに __iselect, __rselect,… などがあったが現在では __merged のみになった。
- 要するにどれも無実な飾り

Catalog

- **photoobj**_mosaic__deepcoadd__merged
  - Forced photometry の結果をまとめたもの
- mosaic_**forceflag_i**__deepcoadd__merged
  - Forced photometry 時のフラグ (Iバンド)
- mosaic_**measlist**__deepcoadd
  - 非 forced photometry の測定結果とフラグ
- mosaic_**measphoto**__deepcoadd
  - Measlist 内のフラックスを物理的な単位に直した

例

```
SELECT
 id, ra2000, decl2000, imag_cmodel
FROM
 s15b_wide.photoobj_mosaic__deepcoadd__merged
WHERE
 tract = 8766;
LIMIT 10;
```

→たいていはゴミしか出てこない
　(フラグをきちんと指定していないから)

フラグの使い方

- フラグを使いたいが Photoobj テーブルにはフラグがない…

(復習)テーブルの結合

- 各生徒を担当する担任が知りたい
- SELECT * FROM studentList JOIN classList ON (studentList.class = classList.class);

studentList

student	class
はな	C組
ひまり	B組
あかり	A組
いちか	C組

classList

class	担任
A組	佐藤
B組	鈴木
C組	高橋

studentList JOIN classList ON (…)

student	class		class	担任
はな	C組	=	C組	高橋
ひまり	B組	=	B組	鈴木
あかり	A組	=	A組	佐藤
いちか	C組	=	C組	高橋

テーブルの結合

- 各天体に対応するフラグが知りたい

  Photoobj JOIN flag USING (id)

photoobj

id	mag
1	23
2	24
3	25
4	26

flag

id	flag
1	NG
2	OK
3	OK
4	NG

mag	Id		id	flag
23	1	=	1	NG
24	2	=	2	OK
25	3	=	3	OK
26	4	=	4	NG

例 (2) [悪い例]

```
SELECT
 id, ra2000, decl2000, imag_cmodel
FROM
 s15b_wide.photoobj_mosaic__deepcoadd__merged
   AS photo
JOIN
 s15b_wide.mosaic_forceflag_i__deepcoadd__merged
   AS iflag
 USING (id)
WHERE
 photo.tract = 8766
 AND iflag.deblend_nchild = 0
 AND NOT iflag.cmodel_flux_flags;
;
```

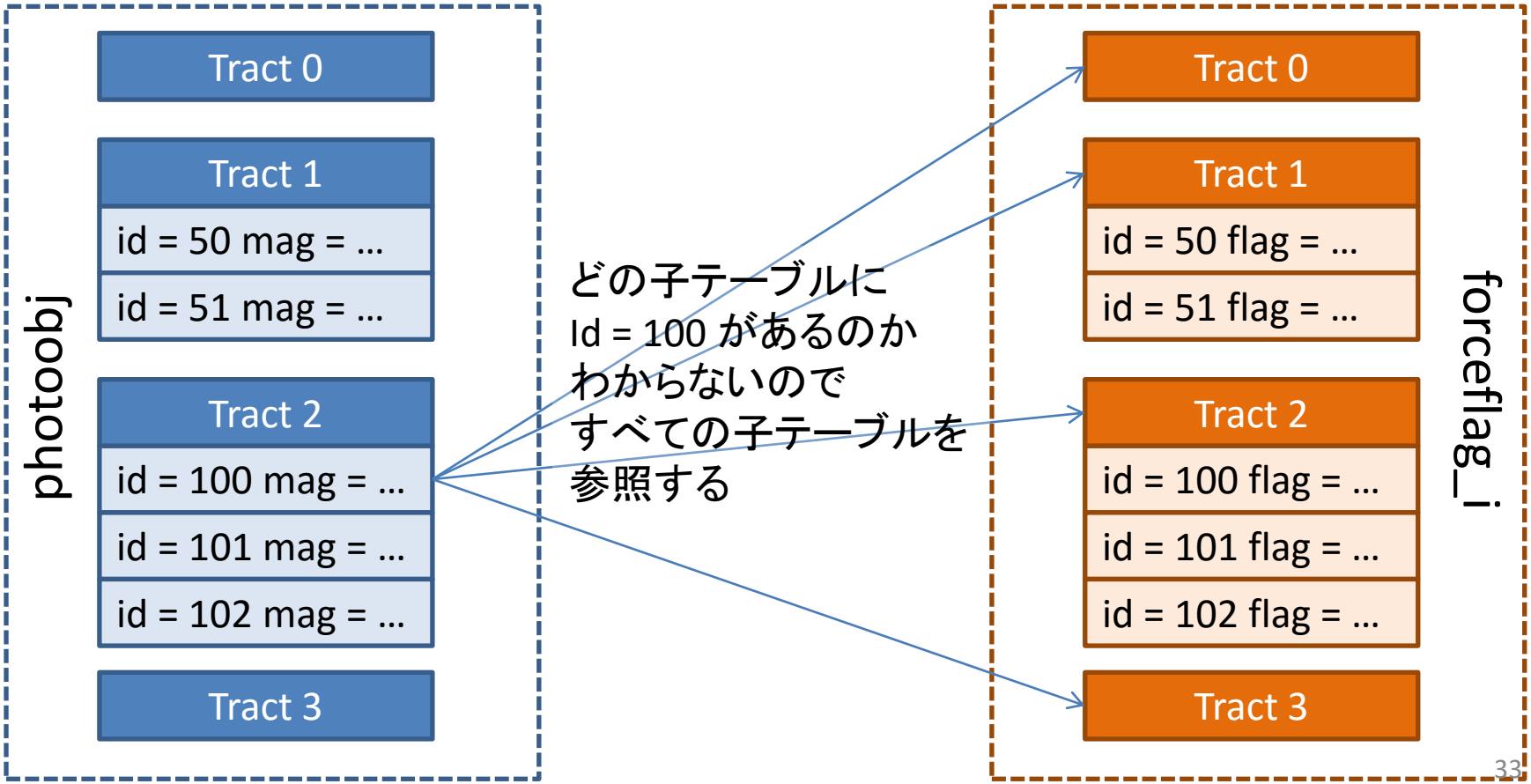
例 (2) [悪い例]

```
SELECT
 id, ra2000, decl2000, imag_cmodel
FROM
 s15b_wide.photoobj_mosaic__deepcoadd__merged
   AS photo
JOIN
 s15b_wide.mosaic_forceflag_i__deepcoadd__merged
   AS iflag
 USING (id)
WHERE
 photo.tract = 8766
 AND iflag.deblend_nchild = 0
 AND NOT iflag.cmodel_flux_flags
;
```

ものすごい遅い

なぜ遅い？

- tract ごとに子テーブルになっているから



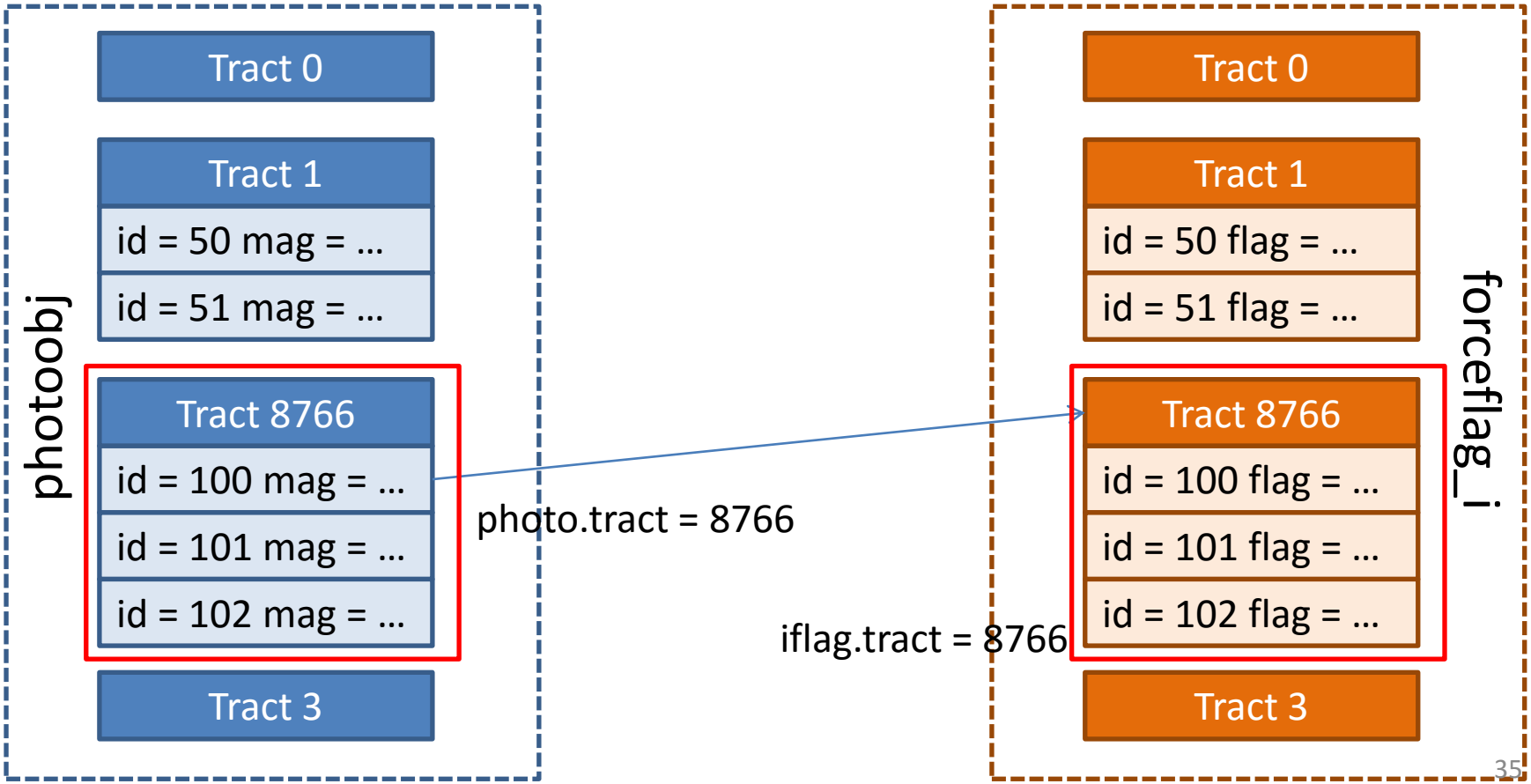
例 (3) [良い例]

```
SELECT
 id, ra2000, decl2000, imag_cmodel
FROM
  s15b_wide.photoobj_mosaic__deepcoadd__merged
    AS photo
 JOIN
  s15b_wide.mosaic_forceflag_i__deepcoadd__merged
    AS iflag
  USING (id)
WHERE
 photo.tract = 8766 AND iflag.tract = 8766
 AND iflag.deblend_nchild = 0
 AND NOT iflag.cmodel_flux_flags;
LIMIT 10;
```

フラグテーブルにも同じ条件を書く

なぜ速い？

- 子テーブルが一つに絞れるから



なお

- このような秘技が不要なものも試験中
- スキーマブラウザで
  - s15b_wide → view → “experimental:….”

まとめ

- SQL についてはググろう
  - MySQL や SQLite や SQLServer ではなく PostgreSQL なので注意
  - 講習会中は何でも気軽に質問してください

- トラクトごとに子テーブルに分かれていることに注意してクエリ文を組み立てよう

実践例

```
SELECT
  phot.id,
  phot.ra2000,
  phot.decl2000,
  phot.iflux_cmodel,
  phot.iflux_cmodel_err,
  phot.iflux_psf,
  phot.iflux_psf_err,
  ref_flag.classification_extendedness,
  phot.a_i

FROM  s15b_wide.photoobj_mosaic__deepcoadd__merged    AS phot
 JOIN s15b_wide.mosaic_forceflag_i__deepcoadd__merged AS i_flag   USING(id)
 JOIN s15b_wide.mosaic_refflag_x__deepcoadd           AS ref_flag USING(id)

WHERE
 phot.tract=8766 AND i_flag.tract=8766 AND ref_flag.tract=8766
 AND NOT i_flag.centroid_sdss_flags
 AND NOT i_flag.flags_pixel_edge
 AND NOT i_flag.flags_pixel_interpolated_center
 AND NOT i_flag.flags_pixel_saturated_center
 AND NOT i_flag.flags_pixel_cr_center
 AND NOT i_flag.flags_pixel_bad
 AND NOT i_flag.cmodel_flux_flags
 AND NOT i_flag.flux_psf_flags
 AND  ref_flag.detect_is_primary ;
```

```
SELECT
  phot.id,
  phot.ra2000,
  phot.decl2000,
  phot.iflux_cmodel,
  phot.iflux_cmodel_err,
  phot.iflux_psf,
  phot.iflux_psf_err,
  ref_flag.classification_extendedness,
  phot.a_i

FROM  s15b_wide.photoobj_mosaic__deepcoadd__merged    AS phot
 JOIN s15b_wide.mosaic_forceflag_i__deepcoadd__merged AS i_flag   USING(id)
 JOIN s15b_wide.mosaic_refflag_x__deepcoadd           AS ref_flag USING(id)

WHERE
 phot.tract=8766 AND i_flag.tract=8766 AND ref_flag.tract=8766
 AND NOT i_flag.centroid_sdss_flags
 AND NOT i_flag.flags_pixel_edge
 AND NOT i_flag.flags_pixel_interpolated_center
 AND NOT i_flag.flags_pixel_saturated_center
 AND NOT i_flag.flags_pixel_cr_center
 AND NOT i_flag.flags_pixel_bad
 AND NOT i_flag.cmodel_flux_flags
 AND NOT i_flag.flux_psf_flags
 AND  ref_flag.detect_is_primary ;
```

```
FROM
s15b_wide.photoobj_mosaic__deepcoadd__merged
    AS phot
```
(メインのテーブル)

```
JOIN
s15b_wide.mosaic_forceflag_i__deepcoadd__merged
    AS i_flag USING(id)
```
(forced photometry フラグ)

```
JOIN s15b_wide.mosaic_refflag_x__deepcoadd
    AS ref_flag USING(id)
```
(レファレンスカタログのフラグ)

```
SELECT
  phot.id,
  phot.ra2000,
  phot.decl2000,
  phot.iflux_cmodel,
  phot.iflux_cmodel_err,
  phot.iflux_psf,
  phot.iflux_psf_err,
  ref_flag.classification_extendedness,
  phot.a_i

FROM  s15b_wide.photoobj_mosaic__deepcoadd__merged    AS phot
 JOIN s15b_wide.mosaic_forceflag_i__deepcoadd__merged AS i_flag   USING(id)
 JOIN s15b_wide.mosaic_refflag_x__deepcoadd           AS ref_flag USING(id)

WHERE
 phot.tract=8766 AND i_flag.tract=8766 AND ref_flag.tract=8766
 AND NOT i_flag.centroid_sdss_flags
 AND NOT i_flag.flags_pixel_edge
 AND NOT i_flag.flags_pixel_interpolated_center
 AND NOT i_flag.flags_pixel_saturated_center
 AND NOT i_flag.flags_pixel_cr_center
 AND NOT i_flag.flags_pixel_bad
 AND NOT i_flag.cmodel_flux_flags
 AND NOT i_flag.flux_psf_flags
 AND  ref_flag.detect_is_primary ;
```

```
SELECT
  phot.id,                        --天体ID
  phot.ra2000,                    --RA
  phot.decl2000,                  --DEC
  phot.iflux_cmodel,              --flux（iバンド）
  phot.iflux_cmodel_err,--その誤差
  phot.iflux_psf,                 --flux（iバンド）
  phot.iflux_psf_err,             --その誤差
  ref_flag.classification_extendedness,
  phot.a_i  --iバンドの吸収    「非-星らしさ」
```

```
SELECT
  phot.id,
  phot.ra2000,
  phot.decl2000,
  phot.iflux_cmodel,
  phot.iflux_cmodel_err,
  phot.iflux_psf,
  phot.iflux_psf_err,
  ref_flag.classification_extendedness,
  phot.a_i

FROM  s15b_wide.photoobj_mosaic__deepcoadd__merged    AS phot
 JOIN s15b_wide.mosaic_forceflag_i__deepcoadd__merged AS i_flag   USING(id)
 JOIN s15b_wide.mosaic_refflag_x__deepcoadd           AS ref_flag USING(id)

 WHERE
  phot.tract=8766 AND i_flag.tract=8766 AND ref_flag.tract=8766
  AND NOT i_flag.centroid_sdss_flags
  AND NOT i_flag.flags_pixel_edge
  AND NOT i_flag.flags_pixel_interpolated_center
  AND NOT i_flag.flags_pixel_saturated_center
  AND NOT i_flag.flags_pixel_cr_center
  AND NOT i_flag.flags_pixel_bad
  AND NOT i_flag.cmodel_flux_flags
  AND NOT i_flag.flux_psf_flags
  AND  ref_flag.detect_is_primary ;
```

```
WHERE
 phot.tract=8766 AND i_flag.tract=8766 AND ref_flag.tract=8766
 AND NOT i_flag.centroid_sdss_flags
```
「座標測定に失敗していない」

```
 AND NOT i_flag.flags_pixel_edge
```
「画像の隅のピクセルを使っていない」

```
 AND NOT i_flag.flags_pixel_interpolated_center
```
「中心付近の重要な箇所に、値を補間されたピクセルがない」

```
 AND NOT i_flag.flags_pixel_saturated_center
```
「中心付近の重要な箇所に、サチったピクセルがない」

```
 AND NOT i_flag.flags_pixel_cr_center
```
「中心付近の重要な箇所に、宇宙線の乗ったピクセルがない」

```
 AND NOT i_flag.flags_pixel_bad
```
「壊れたピクセルを使っていない」

```
 AND NOT i_flag.cmodel_flux_flags
```
「フラックス測定に失敗していない」

```
 AND NOT i_flag.flux_psf_flags
```
「フラックス測定に失敗していない」

```
 AND  ref_flag.detect_is_primary ;
```
「プライマリ天体である」

補遺

- キーワードだけ挙げるので
  詳しくはググってください

SQL演算子

- AND, OR, NOT
  - 命題の「かつ」「または」「でない」
- + , - , *, /, %
  - 四則演算
- a^b
  - 累乗
- <, >, <=, >=
  - 大小比較
- =, !=
  - 等しい、等しくない (C言語と違って ×== ではない)
- *a* BETWEEN *min* AND *max*
  - *min* <= *a* AND *a* <= *max* のこと

- PostgreSQL の演算子は優先順位が直感的でないことがあるので
  こまめに括弧でくくってください

UNION ALL

- テーブルを横につなげるのが JOIN
- テーブルを縦につなげるのが UNION ALL

```
SELECT id from s15b_deep… WHERE …
UNION ALL
SELECT id from s15b_wide… WHERE …;
```

Aggregate functions (集約関数)

- “集約” (カウント・平均など) を行う関数がある
  - SELECT count(id) FROM s15b_wide……;
  - (その他の集約関数についてはググること)

- 集約関数と集約しないコラムは混在できない
  - ×SELECT count(id), flux_cmodel FROM …;

WITH 節

- 一時テーブルを作れる

```
WITH -- 一時テーブルを作って
    tempName1 AS (SELECT … FROM …),
    tempName2 AS (SELECT … FROM …)
-- それを再成形する
SELECT
    …
FROM
    tempName1 JOIN tempName2 ON …;
```