Database

第6回：物理設計

上智大学理工学部情報理工学科
高岡詠子

Schedule

	日程	内容
第1回	10月6日	ガイダンス，データベースとは？
第2回	10月13日	三層スキーマ，データモデル，データベース設計のための仕組み
第3回	10月20日	概念設計：概念モデルとERモデル，論理設計へ
第4回	10月27日	論理設計と正規化
第5回	11月10日	正規化，物理設計
第6回	11月17日	物理設計
第7回	11月24日	SQL言語(データベース定義)
第8回	12月1日	SQL言語(データベース操作：射影制限，結合，合併除の各演算)
第9回	12月8日	SQL
第10回	12月15日	SQL言語(ビュー定義など)
第11回	12月22日	データベース管理システム：トランザクション処理
第12回	1月5日	データベース管理システム：同時実行制御，排他制御
第13回	1月12日	同時実行制御，排他制御，デッドロック
第14回	1月19日	データベース技術動向，リレーショナル代数，まとめ

今日の授業

- 物理設計の手順
- 物理設計のポイント
- 処理効率をあげるための工夫
- テーブル設計に基づく必要ディスクスペース量の計算

データベース設計と3層モデル

対象世界
企業全体

①概念設計（データモデリング）

概念データモデル

②論理設計

論理データモデル

③物理設計

物理データモデル

外部スキーマ

論理性データ独立

概念スキーマ

物理性データ独立

内部スキーマ

データベース設計：実世界をコンピュータシステムに乗せるためのプロセス

物理設計

- 論理設計を実際のシステムの環境に当てはめる
- データベースを実装するための実装設計
- データベースが稼働する環境を設計する
  - DBを実装するためには、DBMSが無理なく動作できる環境を提供する必要がある。

物理設計における作業項目

A

B トランザクション分析

設計要件の整理

製品の機能と制約

・アクセス方式
・格納方式

ポイント

feedback

Bの分析の結果，Aで確認したスペック等で機能的に十分かどうか検討

実際に使用する際の運用やパフォーマンスについての要件を整理する

物理環境の確認

- メインメモリ
- CPUの処理能力（性能）
- Disk 容量
- I/O能力
- ネットワーク構成

物理設計における作業項目

A 物理環境の確認

B

設計要件の整理

製品の機能と制約

・アクセス方式
・格納方式

ポイント

feedback

Bの分析の結果，Aで確認したスペック等で機能的に十分かどうか検討

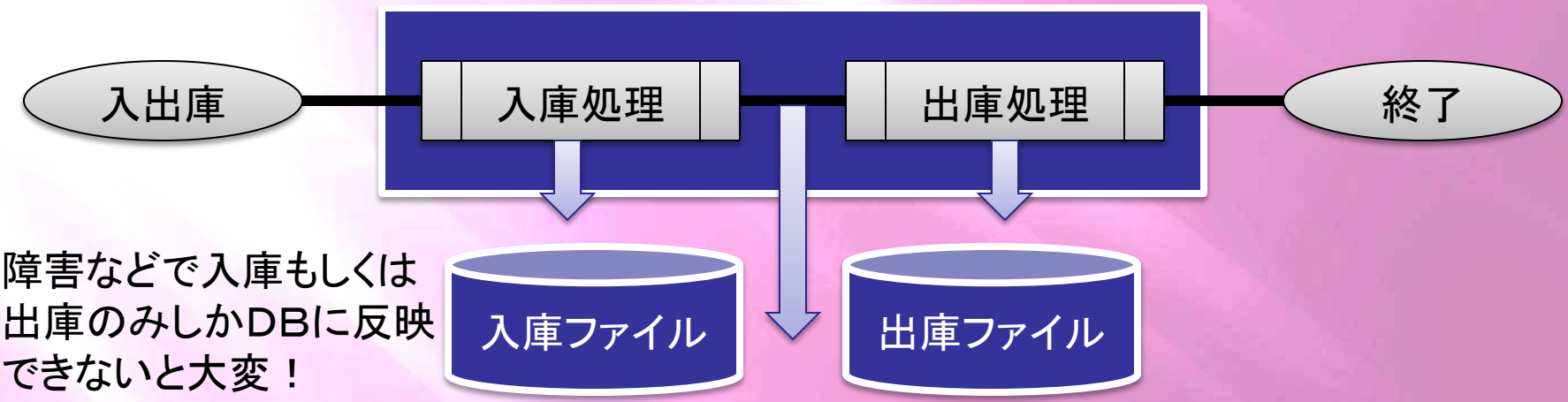
実際に使用する際の運用やパフォーマンスについての要件を整理する

トランザクションとは？

ある企業の在庫管理システムにおける入出庫の処理の一部

仕入れ先から顧客へ直接発送される商品⇒自社で在庫を持たない

入庫と出庫は同時に発生したものとして処理



障害などで入庫もしくは出庫のみしかDBに反映できないと大変！

入出庫の登録は切り離すことができない

このような処理の単位を
という

トランザクション分析

- トランザクションあたりのアクセスデータ量
  - データの　　量や将来の　　量
  - 単位時間当たりの　　や　　の頻度
- データアクセスの形態
  - 　　アクセスが多いか　　アクセスが多いか
  - 　　中心か　　中心か
  - データアクセスの量
- ボトルネック処理
  - 　　データ量が多いか？
  - 時間当たりの　　数が異常に多い？
  - 周期的に大掛かりな　　をする?
  - 多くの表を　　する処理が頻繁?

物理設計における作業項目

A 物理環境の確認

B トランザクション分析

製品の機能と制約

・アクセス方式
・格納方式

ポイント

feedback

Bの分析の結果，Aで確認したスペック等で機能的に十分かどうか検討

実際に使用する際の運用やパフォーマンスについての要件を整理する

設計要件の整理

- の結果からDBに必要な設計要件を整理する
- 物理要件
  - データの　量，将来の　量，　形態，の大きさ等の見積もり
- 運用要件
  - 単位時間あたりの　量
  - タイム
  - 頻度
  - 時の　時間
  - 時の　範囲

Bの分析の結果，Aで確認したスペック等で機能的に十分かどうか検討

物理設計における作業項目

A 物理環境の確認

B トランザクション分析

設計要件の整理

・アクセス方式
・格納方式

ポイント

feedback

Bの分析の結果，Aで確認したスペック等で機能的に十分かどうか検討

実際に使用する際の運用やパフォーマンスについての要件を整理する

製品の機能と制約の確認

- 物理設計の要件がきまった
- データベース製品によって　　や　　が若干異なる
- 選択したデータベース製品について実現可能かどうか確認をする
  - データベース製品の基本機能
    -
  - 可能な　　構造
  - 制限事項：　　の項目数，　　レコード長，格納可能レコード数等の

今日の授業

- 物理設計の手順
- **物理設計のポイント**
- 処理効率をあげるための工夫
- テーブル設計に基づく必要ディスクスペース量の計算

物理設計のポイント

アプリケーションが　　系か　　系か

- 参照系アプリケーションに適した手法
  - をあまり行わない方がよい
  - を使うとよい
  - 集計結果などの項目を
    と毎回計算しなくてすむ
- 更新系アプリケーションに適した手法
  - をすることで，更新時　　を防ぐことができる

正規化

更新時不整合はなくなる代わりに
リレーションが分解されるので
元の情報を得るには
結合しなければならない

アプリケーションの　　の
頻度をレベル分けして　　を
考える！

今日の授業

- 物理設計の手順
- 物理設計のポイント
- 処理効率をあげるための工夫
- 必要ディスクスペース量の計算

非正規化：表を統合

系アプリケーションに適した手法

- 第1正規形を維持して非正規化する
- が主体であればいちいち結合しなくてすむ

履修

履修年度	学生番号	科目コード	科目名	成績
2011	001	5	データベース	A

非正規化：テーブルを分割する

系アプリケーションに適した手法

- 方向への分割
  - 1つの表にアクセスが集中する場合
  - 大量データを保持する表を検索する場合

履修

履修年度	学生番号	学部コード	科目コード	科目名	成績
2011	001	01	5	データベース	A

2011年度履修

年度別にわける

学生番号	学部コード	科目コード	科目名	成績
001	01	5	データベース	A

理工学部履修

学部別にわける

履修年度	学生番号	科目コード	科
2011	001	5	デ

非正規化：項目を追加する

系アプリケーションに適した手法

- **必要な項目をあらかじめ取りこんでおく**

受注品目

受注番号	商品番号	受注個数
0021	001	5

商品

商品番号	商品名	通常価格	割引価格
001	モンドール	5000	4500

購入履歴問い合わせが多い場合や購入時の価格が　　で変わるような場合

受注品目

受注番号	商品番号	商品名	受注個数	購入時価格
0021	001	モンドール	5	5000

非正規化：導出データをあらかじめ計算

系アプリケーションに適した手法

- 集計結果などの項目を
  と毎回計算しなくてすむ

受注品目

受注番号	商品番号	受注個数	購入時価格
0021	001	5	4500

商品

商品番号	商品名	通常価格	割引価格
001	モンドール	5000	4500

受注品目に1行追加されるごとに価格×数量を計算して売り上げ集計表に足しこんでおく

週ごとの売り上げ集計表

週	商品番号	売上合計
11月第一週	001	500,000

インデックス

系アプリケーションに適した手法

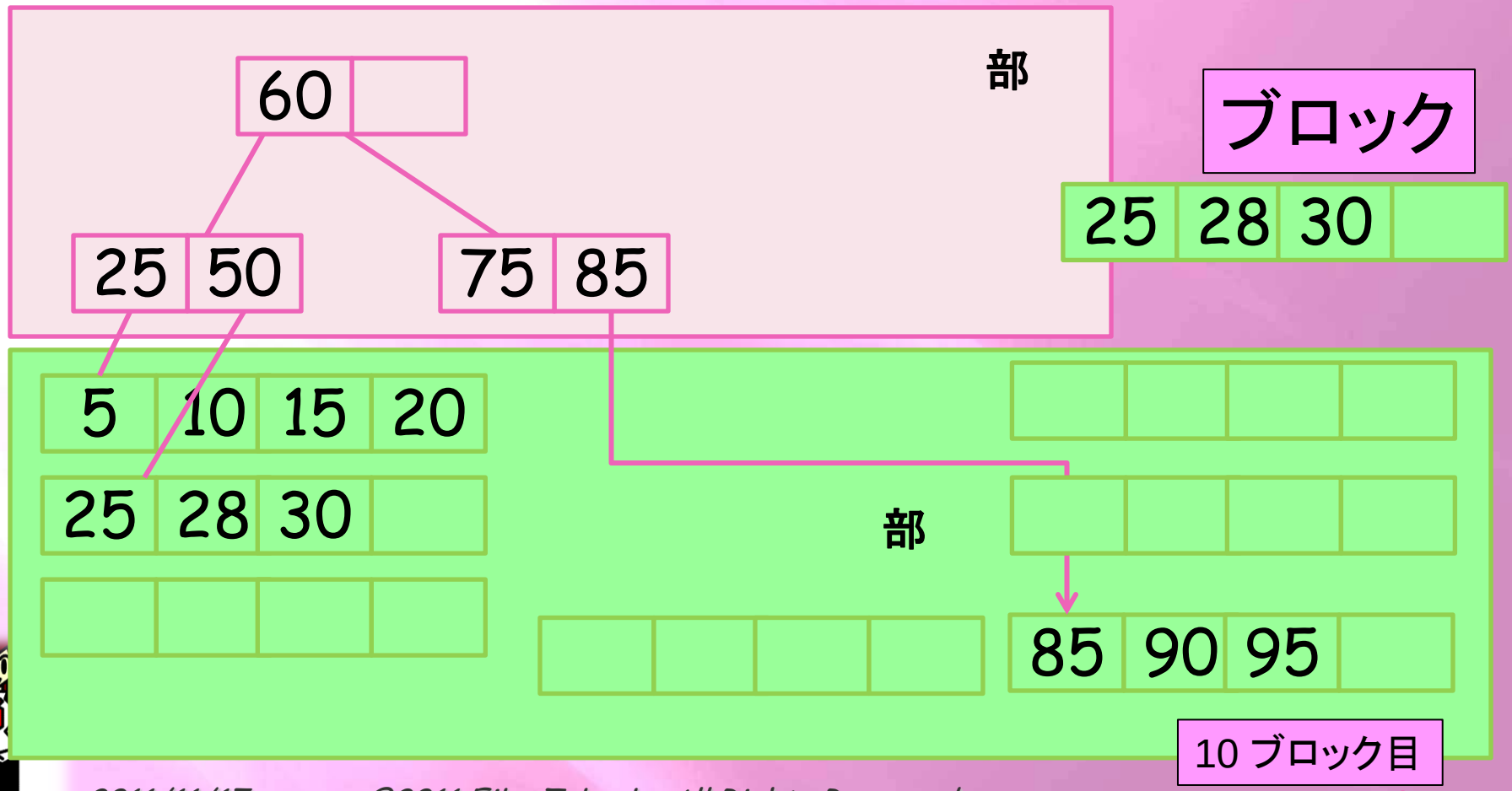
- インデックス設計の考え方
  - 効率は向上する
  - 時にはデータだけでなく も更新する必要がある
  - データ量が ときは設定しないほうが良い
  - のためだけでなくシステム全体としての処理効率の向上を意識し、 最適のためのインデックス定義

なぜインデックスを設定するか

- CPU処理時間：1命令につき　　秒単位
- ディスクのアクセス時間は　　秒単位
- ディスクはCPUに比べて　　倍程度処理が遅い
- 　　をなるべく少なくしたい
- ディスクのアクセスは目的データがたとえ1KBであっても，ブロック単位

B-Tree:RDBMSでは最もポピュラー

- データとキーを完全に分けて管理



B-Treeインデックス

- 95番のデータへのアクセスについて
  - テーブルのブロックサイズ4K
  - 1つのテーブルは1000件のデータ
  - 1ブロックあたり10行のデータを格納できる
  - 合計100ブロックでサイズは400K
  - 検索条件は1000件中1件だけヒットする条件
  - インデックスを使えば：　　ブロック(　　K)
  - インデックスを使わないと：　　ブロック(　　K)

IOが減少するのでパフォーマンスが良い

B-Treeが適する場合

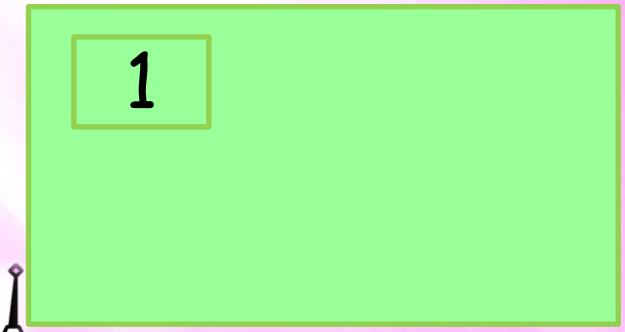
- 全レコードの　　%未満の検索条件
- キー項目の値の種類が　　とき
  - 一般にIDなどで識別される

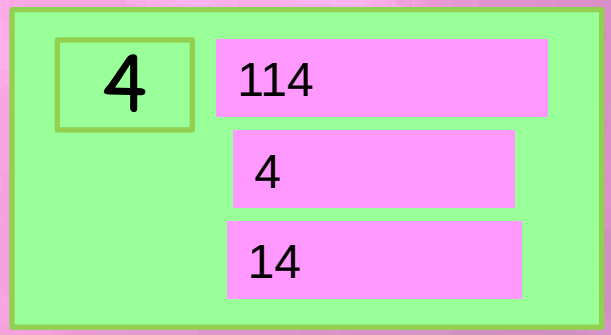
ハッシュインデックス

- B-Treeインデックス：インデックスのルート・ノードだけがキャッシュ・バッファにある
- 実際はディスクアクセスが何度か生じる
- 1回 のアクセスでレコードを取得できるようにしたものがハッシュインデックス
- 　　が一回で済む
- 　　には向かない
- 同じハッシュ値となる値が集中するとよくない
  - 各ページに格納されるレコードの数はなるべく均一になるようにしないとディスクの領域が無駄

ハッシュインデックス

- 受注番号をキーとする
- ハッシュ関数
  - 例えば，受注番号%10を返す（返り値をという）
  - 受注番号114ならば，ハッシュ値は





ビットマップインデックス

- B-Treeインデックスでは効果がえられないときに使う
- 取り得る値の数が　　属性に対して複雑な検索を行う場合に向く
  - 性別，年代等
  - 各列にビットマップインデックスを定義する
- 値の種類の数が多いと領域が多く必要となるので負荷がかかる

学部コードが2で性別がmの人のレコードを抽出

学生番号	学部コード	学科コード	性別
1001	1	11	m
1002	2	22	m
1003	1	12	f
1004	4	43	m
1005	2	21	f
1006	3	31	m

男性	110101
女性	001010

1	101000
2	010010
3	000001
4	000100

110101

AND

010010

010000

参照系・更新系それぞれに適した手法

- 参照系

- 更新系

今日の授業

- 物理設計の手順
- 物理設計のポイント
- 処理効率をあげるための工夫
- テーブル設計に基づく必要ディスクスペース量の計算

テーブル設計

- データ型
- 最大データ長

CHAR	固定長文字列
VARCHAR	可変長文字列
INTEGER	整数
DATE	日付
TIME	時刻
TIMESTAMP	日付と時刻

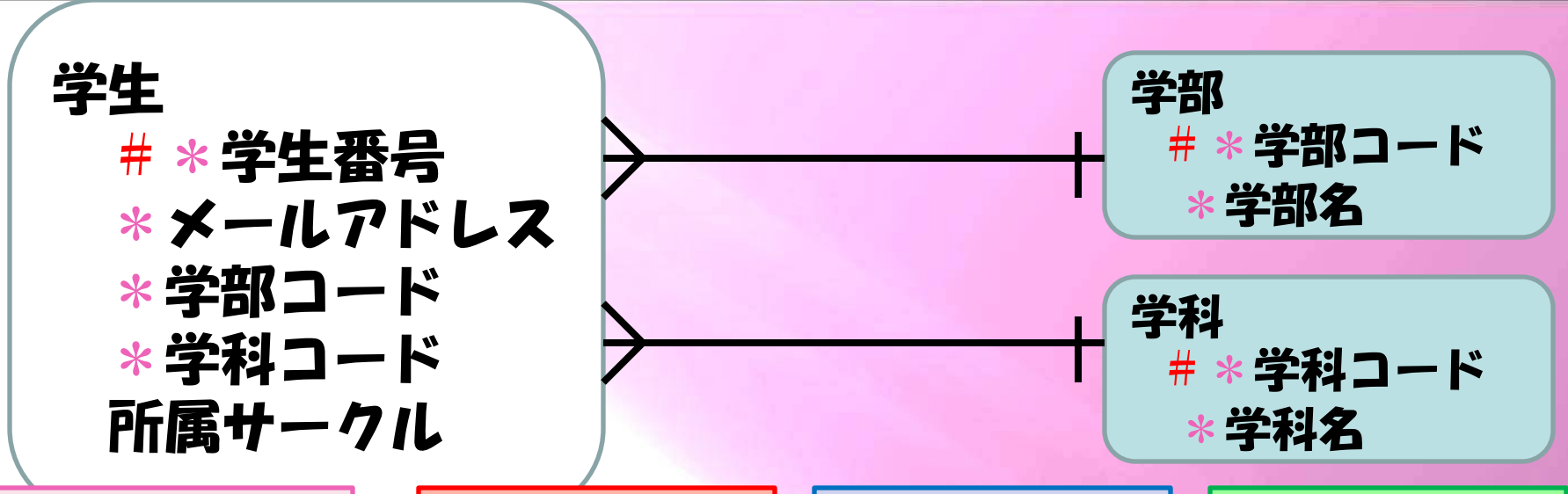
- キー種
- 一意性
- 入力必須 Not Null

制約設計で詳しく！

制約設計

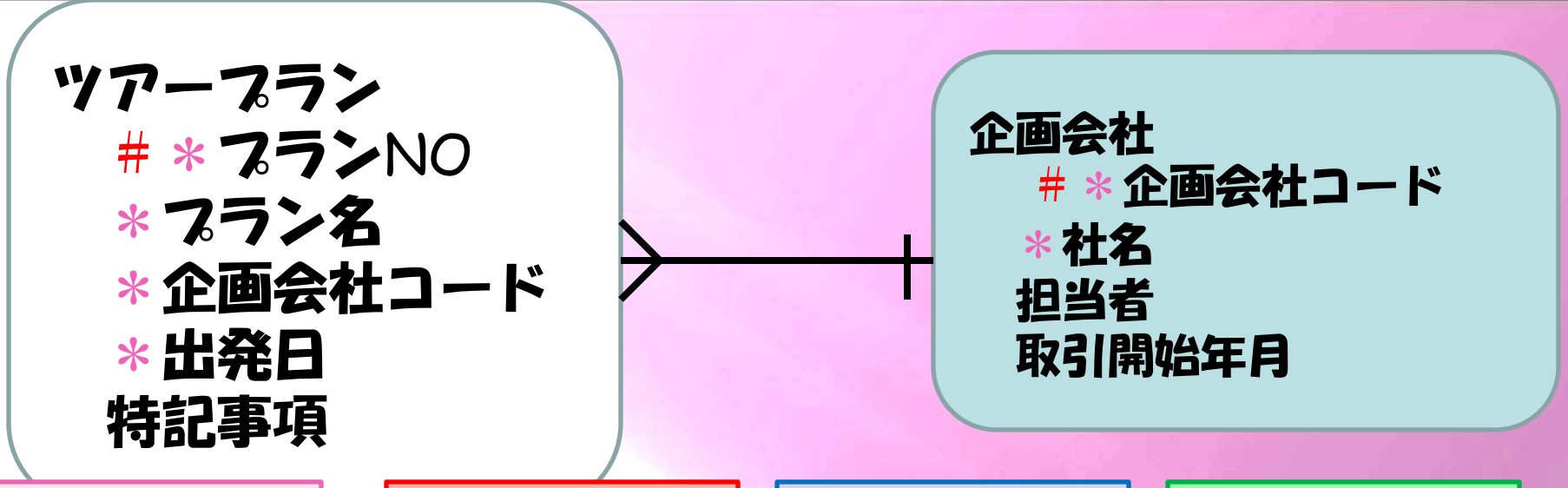
- 制約：値の入力が必ず必要な列に対して定義する(NN1,NN2,NN3….)
- キー制約(PK)
  - 主キー指定(表の中でただひとつ)
  - データの重複は許されない(UNIQUEを含む)
  - NULL値もNG( )
- キー制約：データの重複は許されない
  - すでに他の行の同じ列に存在する値の設定を拒否する
  - 複数の列の組み合わせで一意となる場合は同じ①②など同じ丸付き数字にしておく）
- 制約(FK1,FK2,FK3…)

FKの場合は複数の列が該当することがあるのでFK1,FK2のように番号と一緒に記述する．同時に依存先の表名を書いておく



NOT NULL制約　PRIMARY制約　UNIQUE制約　REFERENCE制約

列名称（属性）	学生番号	メールアドレス	学部コード	学科コード	所属サークル
データ型	INT	VARCHAR	CHAR	CHAR	VARCHAR
最大データ型	12	60	7	7	50
キー種					
一意性					
依存性					
入力必須					
平均データ長	10	20	7	7	10



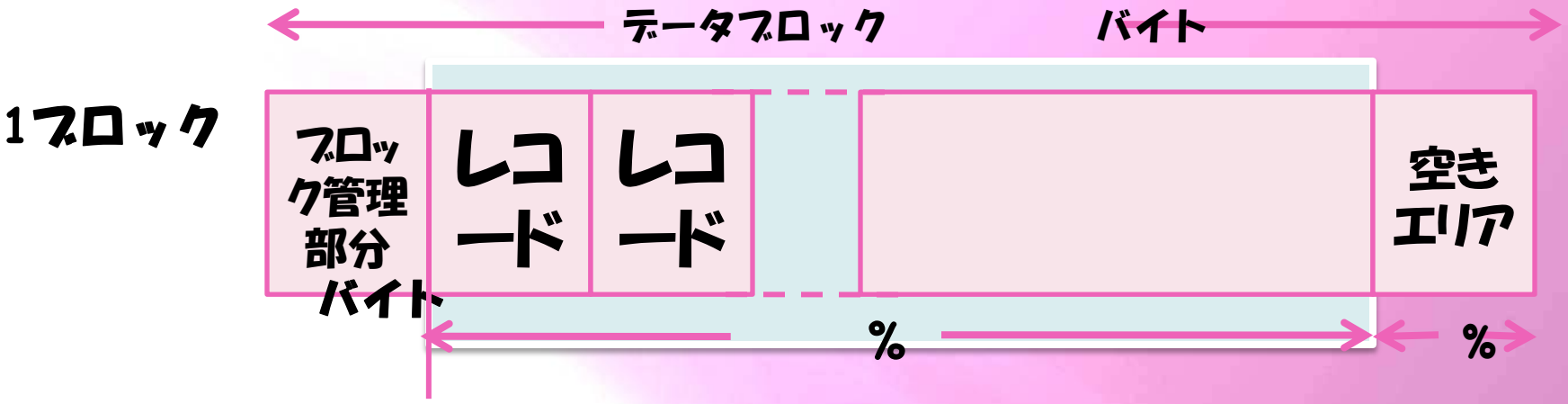
NOT NULL制約　PRIMARY制約　UNIQUE制約　REFERENCE制約

列名称（属性）	プランNO	プラン名	企画会社コード	出発日	特記事項
データ型	INT	VARCHAR	CHAR	DATE	VARCHAR
最大データ型	12	60	7	7	1000
キー種					
一意性					
依存先					
入力必須					
平均データ長	10	45	7	7	300

一つの企画会社には同じプラン名はあってはならない

必要ディスクスペース量の計算

- ツアープランのテーブルインスタンスチャートから表の容量を見積もりなさい．各手順での計算値の小数点以下の取り扱いに気をつけること．用件は以下の通り
  - テーブル行数：80000
  - PCTFREE　10%
  - ブロックサイズ 4K(4096バイト)
  - ブロック管理部分　64バイト
  - 列長はオーバーヘッド域を考えず，平均データ長を足しこんだものとする



- 1ブロックあたりのデータ用使用可能領域のバイト数
  1ブロックあたりの使用可能領域のバイト数×(1-PCTFREE)
- 1ブロックあたりの格納行数
  列長=
- 必要な通常ブロック数
  テーブル行数
- テーブルスペースの必要な容量(Kバイト単位)